UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CIÊNCIA DA COMPUTAÇÃO

CONTRIBUIÇÕES À PLATAFORMA BOINC PARA COMPUTAÇÃO VOLUNTÁRIA EM DISPOSITIVOS MÓVEIS COM ANDROID

TRABALHO DE GRADUAÇÃO

Hugo Stefan Kaus Puhlmann

Santa Maria, RS, Brasil

2013

CONTRIBUIÇÕES À PLATAFORMA BOINC PARA COMPUTAÇÃO VOLUNTÁRIA EM DISPOSITIVOS MÓVEIS COM ANDROID

Hugo Stefan Kaus Puhlmann

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de

Bacharel em Ciência da Computação

Orientador: Profª. Dra. Andrea Charão

Trabalho de Graduação N. 362
Santa Maria, RS, Brasil

2013

Universidade Federal de Santa Maria
Centro de Tecnologia
Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

CONTRIBUIÇÕES À PLATAFORMA BOINC PARA COMPUTAÇÃO VOLUNTÁRIA EM DISPOSITIVOS MÓVEIS COM ANDROID

elaborado por
Hugo Stefan Kaus Puhlmann

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Andrea Charão, Dra.
(Presidente/Orientador)

Benhur Stein, Dr. (UFSM)

Patrícia Pitthan, Dra. (UFSM)

Santa Maria, 26 de Julho de 2013.

“And if your head explodes with dark forebodings too, I’ll see you on the dark side of the moon.”

— ROGER WATERS

RESUMO

Trabalho de Graduação
Ciência da Computação
Universidade Federal de Santa Maria

CONTRIBUIÇÕES À PLATAFORMA BOINC PARA COMPUTAÇÃO VOLUNTÁRIA EM DISPOSITIVOS MÓVEIS COM ANDROID

AUTOR: HUGO STEFAN KAUS PUHLMANN
ORIENTADOR: ANDREA CHARÃO
Local da Defesa e Data: Santa Maria, 26 de Julho de 2013.

A utilização de recursos computacionais em pesquisas científicas é muito comum nos dias atuais, pois estes permitem a realização de grandes cálculos de maneira rápida, especialmente se comparados à velocidade que os mesmos processos seriam feitos por um ser humano. O BOINC é um *middleware* para computação de grade ou voluntária que oferece suporte a populares sistemas operacionais voltados para *desktops*. Todavia, a plataforma não oferece suporte integral ao Android, sistema operacional popular em dispositivos móveis, que tornam-se interessantes como fonte alternativa de voluntários principalmente pela combinação de dois fatores: o rápido aumento do número de dispositivos vendidos com plataforma instalada e o poder bastante expressivo de processamento alcançado por aparelhos atuais. Este projeto tem como objetivo dar continuidade ao trabalho de adição de suporte ao Android no BOINC, colaborando no desenvolvimento de duas funcionalidades ainda não desenvolvidas: tratamento de interrupções no processamento de projetos causadas pela atualização do pacote de aplicação, e o oferecimento de uma opção que permita ao usuário configurar se deseja armazenar dados de projeto em um cartão *SD*. Foi possível concluir a execução ta tarefa de gerenciamento de interrupções, sendo esta integrada ao projeto. O suporte à utilização de cartões *SD* foi parcialmente concluído, não sendo integrado ao código do projeto.

Palavras-chave: Android. BOINC. Computação Voluntária.

ABSTRACT

Undergraduate Final Work
Undergraduate Program in Computer Science
Federal University of Santa Maria

CONTRIBUTIONS TO BOINC PLATFORM OF VOLUNTEER COMPUTING ON MOBILE DEVICES

AUTHOR: HUGO STEFAN KAUS PUHLMANN
ADVISOR: ANDREA CHARÃO
Defense Place and Date: Santa Maria, July 26th, 2013.

The use of computational resources for scientific research has become very common today, since it allows large calculations to be performed quickly, especially when compared to the time it would take for a human to perform them. *BOINC* is a *middleware* specialized for grid and volunteer computing that currently supports the most popular operating systems for desktop computers. However, the platform does not provide full support for Android, an operating system very popular on mobile devices, which are becoming an interesting alternative source of volunteers, due to the combination of two main factors: the increasing number of devices sold running the platform and the significant computing power reached by newly released devices. This project aims to continue the work on adding full Android support to *BOINC* by contributing to de development of two unimplemented features: the handling of interruptions to project computations caused by application package updates, and the addition of an option to allow the user to set whether he wants to store projects data on SD Cards or not. The task of handling the interruption caused by package updates was successful, being integrated into the project's code. The support to *SD* cards was partially concluded, not integrated into the official repository yet.

Keywords: Android, BOINC, Volunteer Computing.

LISTA DE FIGURAS

LISTA DE APÊNDICES

SUMÁRIO

1 INTRODUÇÃO

A computação de dados desempenha papéis importantes em avanços de âmbito científico, desde ferramentas primitivas como o ábaco até computadores digitais utilizados atualmente. Computação, em sua definição, consiste na execução de procedimentos sobre um determinado conjunto de dados, com a finalidade de obter-se possíveis conclusões a respeito de tal conjunto. Tais procedimentos podem requirir diferentes quantidades de tempo para sua execução, de acordo com sua complexidade e com o poder computacional disponível.

Para procedimentos de alta complexidade com restrições de tempo, pode ser necessário um grande poder computacional, o que, em termos de computadores digitais, pode ser obtido de duas formas básicas: através da distribuição de tarefas entre diversos dispositivos ou com a aquisição direta de hardware com maior poder computacional. Em diversas situações o custo para compra de hardware pode inviabilizar a execução de projetos, o que torna a computação distribuída, nas suas mais variadas formas, uma possível alternativa para obter-se maior poder computacional.

Um sistema distribuído consiste em diversas entidades computacionais autônomas (nodos) executando programas distribuídos e comunicando resultados de alguma forma. De acordo com a natureza de um problema qualquer a ser resolvido, pode-se desenvolver algoritmos que tomem proveito dos diversos nodos inter-conectados pelo sistema, o que pode acelerar a completude de um processamento. Todavia, a aquisição destes dispositivos envolve custo financeiro, o que pode ser um fator impossibilitante.

Em uma alternativa de viabilização de projetos de científicos, pode-se utilizar sistemas de computação voluntária, dentre os quais destaca-se o BOINC, que oferece suporte a funcionalidades para execução de tarefas de projetos de computação voluntária, com versões disponíveis para diversos sistemas operacionais, através da instalação de uma aplicativo único (cliente) em computadores voluntários. Todavia, um dos principais sistemas voltados para computação móvel, o Android, ainda não conta com um cliente completamente desenvolvido, não sendo disponibilizados pacotes de aplicação para instalação nesta plataforma.

1.1 Objetivo Geral

Encontra-se em estado avançado de desenvolvimento pela equipe do BOINC um cliente BOINC para Android, sendo o objetivo deste trabalho é continuar o desenvolvimento de fun-

cionalidades restante do cliente BOINC para Android, tendo como principal meta minimizar o esforço requerido por parte de desenvolvedores de projeto, de modo a facilitar a implementação de suporte a voluntários usuários da plataforma Android. O sistema operacional foi escolhido como plataforma alvo deste trabalho pois trata-se de um sistema que oferece maior liberdade para a construção de aplicações, especialmente quando comparado ao seu principal concorrente de mercado nos dias atuais (PITKANEN, 2008).

Deseja-se implementar funcionalidades já disponíveis nas versões já estabelecidas dos clientes para os sistemas operacionais oficialmente suportados (Windows, Mac OS X, GNU/Linux), salvo eventuais limitações de sistema.

Também é essencial adicionar funcionalidades de gerenciamento de questões específicas de dispositivos móveis, como agendamento de unidades de trabalho de acordo com a temperatura e nível de carga da bateria do dispositivo, limitações de recursos computacionais disponíveis nesta categoria de dispositivos e considerações a respeito de tráfego de dados.

1.2 Justificativa

Dado que sistemas de computação voluntária tendem a escalar seu poder de processamento de acordo com o número de computadores disponíveis para execução de suas tarefas, é de grande importância maximizar a quantidade de plataformas suportadas por esses sistemas, a fim de aumentar o número de potenciais voluntários.

O Android é um sistema operacional lançado em 2007, baseado em Linux e projetado para atender necessidades de dispositivos móveis como *smartphones* e *tablets*. No primeiro trimestre de 2013, foram vendidos em aproximadamente 426 milhões de telefones celulares, dos quais cerca de 74.4% utilizavam o Android (RIVERA; MEULEN, 2013), tornando a plataforma uma fonte de voluntários com grande potencial quantitativo. O poder de processamento de tais dispositivos alcança patamares elevados, podendo-se citar como exemplo o smartphone LG Nexus 4, que utiliza o sistema Qualcomn SnapDragon S4 APQ8064, que inclui um processador de quatro núcleos independentes capazes de alcançar uma frequência de ciclo de 1700 MHz, decodificando até três instruções por ciclo (QUALCOMN, 2013).

A popularização dos *smartphones* equipados com *Systems on a Chip (SoC)* torna cada vez mais comum dispositivos com processadores multi-núcleos, poderosas unidades de processamento gráfico, capazes de executar bilhões de operações de ponto flutuante por segundo. Dado que, assim como em computadores tradicionais, em uma considerável parcela de tempo

esse poder de processamento é desperdiçado durante períodos de ociosidade, torna-se interessante utilizar estes dispositivos para realizar pesquisas científicas (EASTLACK, 2011).

O BOINC é uma plataforma amplamente utilizada por projetos de computação voluntária e suportada pelos principais sistemas operacionais *desktop*. Porém, ainda não há suporte completo ao Android, o que acaba impossibilitando usuários do sistema de voluntariarem-se em tais projetos. Esse fato desperdiça uma fonte de voluntários de grande potencial, dada a popularização massiva do sistema nos últimos anos, muitos destes capazes de grande poder de processamento devido à utilização de avançados *SoC*. Tal desperdício de potencial torna justificável esta pesquisa, que visa possibilitar que usuários do Android doem parte do processamento de seus dispositivos em prol de projetos científicos.

2 REVISÃO BIBLIOGRÁFICA

Para o desenvolvimento deste trabalho, foram necessários estudos de conceitos tangentes ao desenvolvimento do cliente BOINC para Android. Conduziu-se inicialment um estudo sobre computação voluntária, com a finalidade de identificar aplicação do conceito à *smartphones*. Para possibilitar o reconhecimento e implementação de funcionalidades necessárias ao cliente, foi necessário compreender conceitos do BOINC e fundamentos técnicos de desenvolvimento Android. Por fim, elaborou-se estratégias possíveis para a utilização do cliente no Android, tendo sido estas abandonadas mediante estudo de uma aplicação anteriormente existente.

2.1 Computação Voluntária

A computação voluntária assemelha-se ao processamento em grade, porém as máquinas envolvidas não são necessariamente propriedade de uma única entidade e, portanto, é uma alternativa menos custosa financeiramente.

A ideia central da computação voluntária é oferecer um mecanismo de computação paralela de alto desempenho de forma rápida e barata, permitindo que voluntários doem parte do poder de processamento de seus computadores para projetos de seu interesse (SARMENTA, 2001).

2.2 BOINC

O *Berkeley Open Infrastructure for Network Computing (BOINC)* é um *middleware* para computação distribuída de grade ou voluntária, originalmente criado para ser base do projeto SETI@Home, e atualmente é uma plataforma para aplicações das mais diversas áreas. O *BOINC* oferece suporte a três populares sistemas operacionais: GNU/Linux, Mac OS X e Windows. A plataforma é amplamente difundida em projetos de computação voluntária. Sua popularidade deve-se também ao fato de possuir uma arquitetura extensível e seu *framework* de aplicação ter sido projetado para possibilitar que projetos existentes sejam utilizados na plataforma.

A plataforma inclui características como autonomia de projetos, flexibilização para voluntariado, *framework* de aplicação flexível, métodos de proteção contra diversos tipos de ataques, servidor escalável de alta performance, bem como arquitetura extensível e de código

aberto.

Projetos baseados na plataforma *BOINC* são entidades independentes, *i.e.* projetos não são afetados pelo estado de outros, sendo que cada um destes contém suas próprias aplicações, banco de dados, web site e servidor(es) (figura 2.1). A arquitetura de projetos baseados no BOINC pode ser entendida como um série de camadas:

1. Camada *back-end*, responsável pelo controle do projeto, bem como armazenamento e triagem de dados recebidos das tarefas executadas por voluntários;

2. Camada de exposição do *back-end* do projeto, oferecendo um servidor *web* com uma série de arquivos que serão acessados por voluntários e responsável pelo recebimento dos dados enviados por participantes e encaminhamento destes ao *back-end*;

3. Camada do voluntário, componentes localizados na máquina dos voluntários do projeto, sendo responsável pelo processamento das aplicações científicas e o envio da resposta à camada de exposição.

Em termos de implementação, um projeto consiste em uma hierarquia de diretórios contendo dados necessários para sua execução, bem como um banco de dados MySQL para armazenamento de informações (ANDERSON; MAIRE, 2011). Uma aplicação bastante popular é o *SETI@Home*, que faz uso deste tipo de arquitetura para permitir que usuários colaborem com a análise de ondas de rádio de banda baixa, a fim de identificar ondas incomuns que podem ser indício de vida extraterrestre (ANDERSON et al., 2009).

2.2.1 Conceitos Básicos

O BOINC sustenta sua arquitetura em torno de uma série de entidades co-relacionadas, onde cada entidade é armazenada em tabelas próprias no banco de dados MySQL associado ao projeto. Abaixo, uma breve definição das entidades utilizadas na arquitetura da plataforma (ANDERSON, 2008).

1. Aplicação: consiste em diversos programas de computador (especificamente compilados para as plataformas suportadas pelo projeto) e unidades de trabalho, sendo importante notar que cada projeto pode conter diversas aplicações.

2. Plataforma: sistema para qual um programa é compilado, sendo geralmente identificada pela combinação da arquitetura de CPU e sistema operacional.

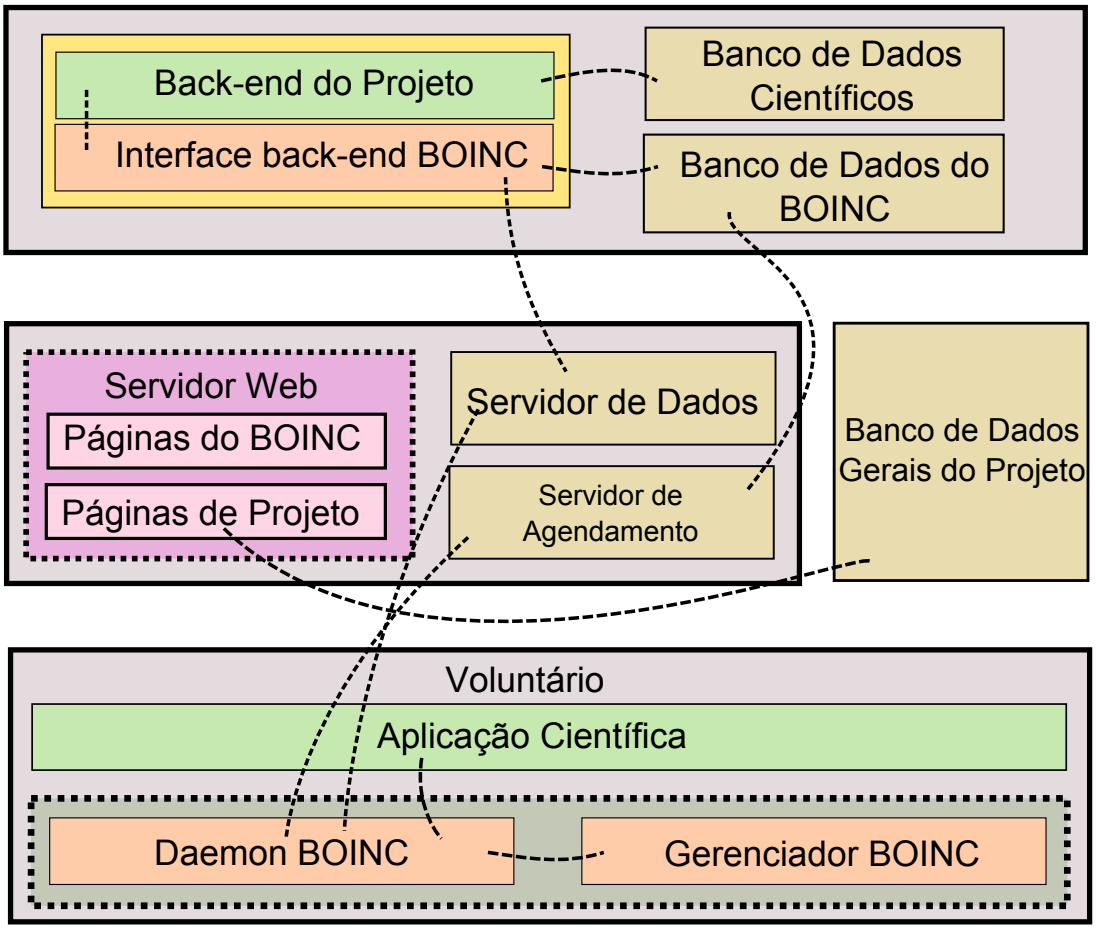


Figura 2.1: Arquitetura Geral de Projetos Baseados no BOINC

3. Versões de Aplicação: aplicações podem apresentar diversas versões, onde cada versão compilada para uma plataforma específica é chamada versão de aplicação.

4. Unidades de Trabalho: são tarefas a serem executadas, podendo estas incluírem um número virtualmente ilimitado de arquivos de entrada. Unidades de trabalho são associadas a uma aplicação, sem levar em consideração suas versões e plataformas, podendo conter especificações como requisitos mínimos de recursos e tempo limite para submissão de resultados.

5. Resultado: é definido como o produto de uma determinada computação associada a uma unidade de trabalho.

6. Cliente: programa executado por cada voluntário de projeto, sendo este baixado e instalado manualmente, sendo responsável pela anexação e gerenciamento de projetos.

2.2.2 Cliente BOINC

O cliente BOINC é um componente da infraestrutura da plataforma, sendo responsável pelo gerenciamento das questões envolvidas na execução de projetos que utilizem a plataforma. Dentre as principais responsabilidades do cliente encontra-se a intermediação de comunicações remotas entre a máquina executora e servidores de projeto, realizando ações como o *download* de arquivos de entrada para cada tarefa e executáveis necessários para a realização das operações de cada projeto. Também é de responsabilidade do cliente o envio dos resultados de tarefas ao servidor do projeto.

O componente é incluído em todos pacotes de instalação disponíveis para os sistemas operacionais suportados, apresentando-se de maneiras ligeiramente diferentes em cada sistema operacional. Em sistemas baseados em *UNIX* o cliente é geralmente instalado como um *daemon*, enquanto na versão para Windows pode tanto ser instalado como uma aplicação comum, quanto um *service* do sistema.

É importante ressaltar que o cliente não é apresentado ao usuário de maneira direta, pois não é controlável diretamente pelo voluntário. O controle é realizado através do componente *boinccmd*, que comunica-se com o cliente utilizando chamadas remotas, permitindo que mantenha-se uma rígida separação entre funcionalidades de controle do cliente e as operações executadas pelo mesmo. Também são fornecidas junto com o pacote de aplicação interfaces gráficas para controle do cliente, que permitem visualizar informações de tarefas sendo executadas, transferências realizadas, adição de novos projetos, configuração do BOINC, etc (figura 2.2).

2.3 Android

O Android é um sistema operacional que foi desenvolvido inicialmente pela Android, Inc. (ALLIANCE, 2007) e adquirido em 2005 pela Google, Inc. (WEEK, 2005), sendo baseado em Linux e primeiramente voltado para dispositivos móveis (ALLIANCE, 2007).

A plataforma oferece um *Software Development Kit*, isto é, um conjunto de bibliotecas e programas, de código aberto e gratuito, tornando a plataforma bastante acessível para desenvolvedores.

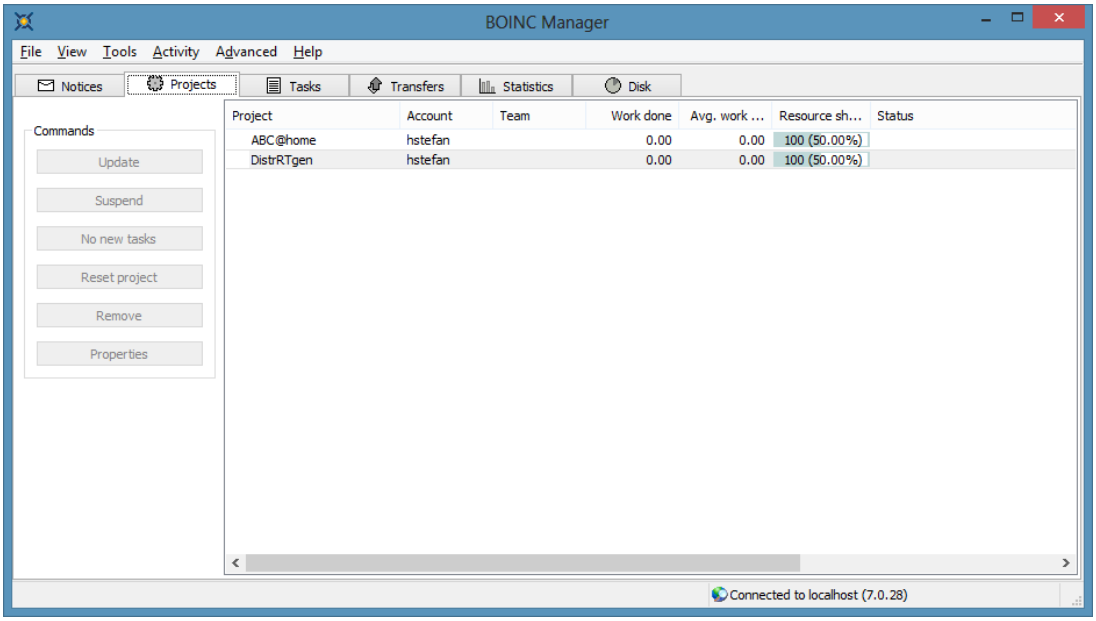


Figura 2.2: Interface Gráfica para Gerenciamento do Cliente

2.3.1 Android Manifest

Cada aplicação desenvolvida para o Android conta com um arquivo *XML*, AndroidManifest.xml em seu diretório raiz, sendo este uma representação de propriedades básicas e essenciais da aplicação, como o pacote Java, seus diversos componentes (activities, services, broadcast receivers, etc), permissões requisitadas ao usuário instalador da aplicação, versão mínima do *SDK*, etc (GOOGLE, 2013c).

2.3.2 Activities

Uma *Activity* é um componente de aplicação que possibilita interação com o usuário, a fim de permitir que o mesmo utilize-se das funcionalidades da aplicação, podendo esta possuir múltiplas *activities*.

Activities são representadas pelo *SDK* do Android através da classe abstrata *Activity*, que define uma série de métodos para controle do ciclo de execução da aplicação, destacando-se os métodos *onCreate* e *onPause*, onde o primeiro é invocado quando a atividade é criada pelo sistema e o segundo chamado sempre que esta for abandonada pelo usuário (GOOGLE, 2013a). O ciclo de vida de uma *Activity* pode ser melhor compreendido através da figura 2.3.

A definição dos elementos de interface é geralmente feita através de um arquivo *XML*, com o uso de *tags* que identificam cada tipo de elemento (*view*) pré-definido pelo *XML scheme* de *layout* no Android, onde cada elemento declarado possui valor identificador único para pos-

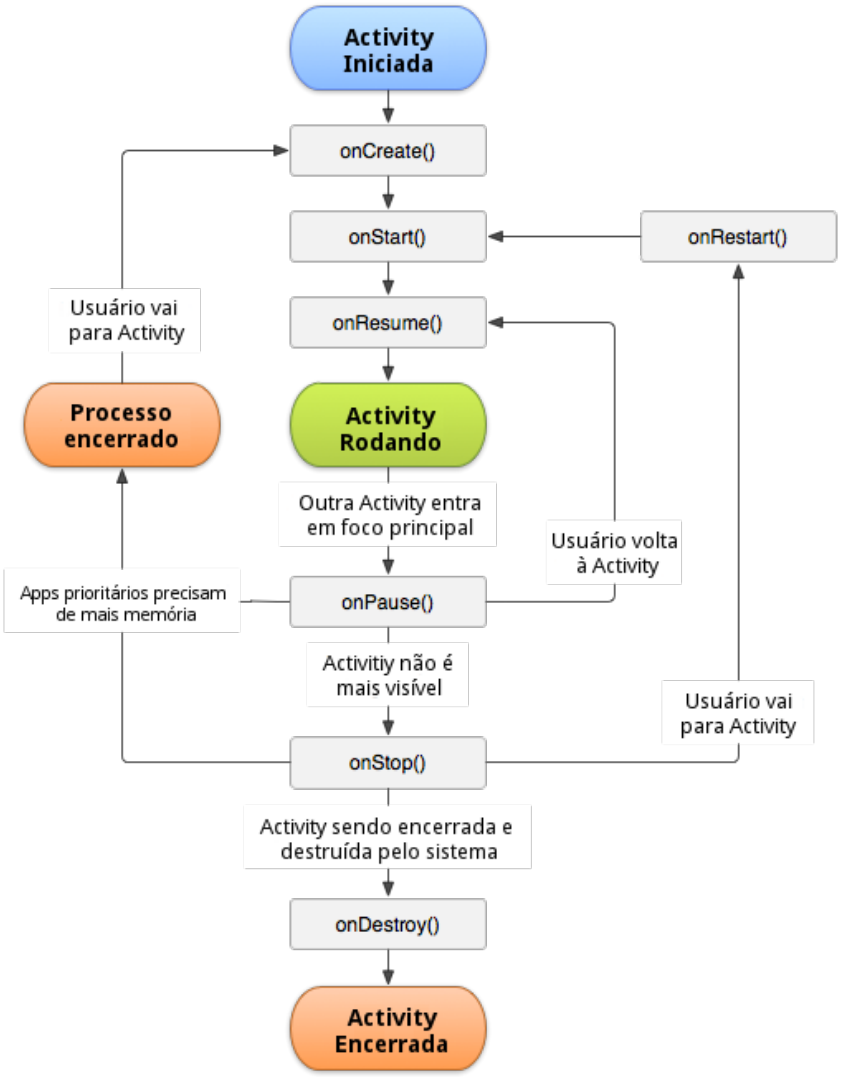


Figura 2.3: Ciclo de Vida de Activities no Android. Fonte: (GOOGLE, 2013a)

sibilitar o acesso dinâmico de propriedades do mesmo, através do método *getResourceById*.

2.3.3 Services

Services são componentes de aplicação geralmente utilizados para processamento de longas ações, de forma que sua execução independe da aplicação estar no plano principal de operação, a partir do ponto onde um componente de aplicação o inicia através do comando *startService*. *Services* são uma alternativa para realizar processamento em plano secundário, uma vez que estes têm ciclo de vida independente de quaisquer outros componentes da aplicação, podendo permanecer ativos mesmo após esta ser destruída pelo sistema (GOOGLE, 2013b).

Também é possível utilizar-se de protocolos definidos com *Android Interface Design Language (AIDL)* para expor funcionalidades do serviço, tornando viável que outros compo-

nentes da aplicação conectem-se ao serviço para realizar comunicação entre os componentes. Através da troca de mensagens descritas em *AIDL*, é possível realizar tanto o envio quanto o recebimento de objetos serializáveis (GOOGLE, 2013b).

Services podem apresentar-se de duas formas: sendo iniciados por um componente qualquer através do método *startService*, ou conectados a um componente utilizando-se o método *bindService*, de forma a realizar comunicação entre as duas partes. Ambos os casos apresentam ciclos de execução distintos, como ilustrado na figura 2.4 (GOOGLE, 2013b).

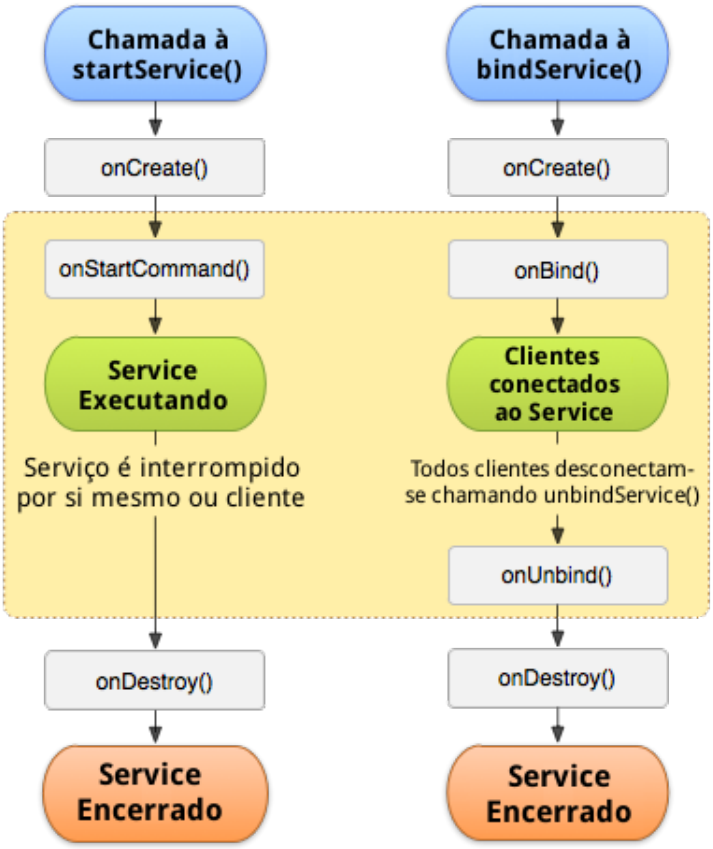


Figura 2.4: Ciclo de Vida de Services no Android. Fonte: (GOOGLE, 2013b)

2.3.4 Broadcast Receivers

Um *Broadcast Receiver* consiste em um objeto destinado a gerenciar um ou mais eventos que são propagados pelo sistema, podendo estes serem de escopo global, com particularidades e técnicas de implementação diferenciadas. Diferente de *Activities* e *Services*, possuem ciclo de vida simplificado, uma vez que só representam objetos válidos dentro do escopo da função *onReceive*, sendo destruídos logo após o término da execução deste método. Isto acarreta também

algumas limitações, como a impossibilidade de uma conexão direta com um *Service* e o fato de não serem permitidas quaisquer operações assíncronas, uma vez que não será possível retornar qualquer valor ao objeto, pois este constitui instância válida somente durante a execução do método *onReceive*.

- Escopo global: Implementados como uma subclasse de *BroadcastReceiver*, sendo utilizados preferencialmente para a comunicação de eventos entre aplicações diferentes, incluindo eventos originados pelo próprio sistema operacional. Também são capazes de receber eventos locais, porém esta estratégia pode mostrar-se menos eficaz que a seguinte, devido a utilização de protocolos de comunicação inter-processual ao invés de chamadas de métodos localmente (GOOGLE, 2013d).

- Escopo local: Implementados como subclasse de *LocalBroadcastReceiver*, sendo eficazes para o controle de eventos locais, isto é, aqueles disparados pela aplicação e destinados a componentes da mesma. Se mostram mais eficientes em termos de custo computacional, pois, ao contrário de *Broadcast Receivers*, não dependem de protocolos de comunicação entre os processos envolvidos, o que é benéfico para a eficiência no controle de eventos locais, mas inviabilizante para gerenciamento daqueles oriundos de outras aplicações (GOOGLE, 2013e).

Esses componentes se mostram especialmente úteis à medida que oferecem um meio de receber eventos disparados pelo sistema, possibilitando, por exemplo, notificações de remoção de cartão de memória, carga de bateria atingindo níveis baixos, alterações em pacotes instalados, etc. Pode-se registrar *Broadcast Receivers* de duas formas no Android, através do método *registerReceiver* (definido para objetos do tipo *Context*), ou ainda, estaticamente através do arquivo *AndroidManifest.xml*, abordado nas seções anteriores.

2.3.5 Intents e Intent Filters

Um objeto do tipo *Intent* representa um conjunto de informações de interesse do objeto recebedor de um *Intent*, e/ou de interesse do próprio sistema operacional, contendo, principalmente:

- Nome do componente que irá tratar o *Intent*, sendo esta informação representada por um objeto do tipo *ComponentName*, um conjunto de dados composto pelo nome do pacote java da aplicação e do próprio componente.

- Ação representada pelo *Intent*, sendo essa um objeto do tipo *String*.
- Dados aos quais deve ser aplicada a ação, sendo representados por um *Uniform Resource Identifier (URI)* correspondendo ao tipo do dado envolvido e o tipo *MIME* do mesmo.
- Categoria do *intent*, definida através de uma *String*, representando o tipo de componente que deve tratar o *intent*.
- Pares chave-valor para envio de dados adicionais que devem ser entregues ao recebedor; por exemplo, para um *intent* contendo uma ação relacionada a datas, poderia-se incluir informações de fuso horário no espaço destinado para dados adicionais.
- Marcadores ou *Flags*, utilizados principalmente para informar o sistema operacional do modo que *intent* deve ser processado.

Intents são os mecanismos ativadores para os três componentes anteriores, sendo a ativação destes realizada através da declaração de *Intent Filters*, isto é, filtros que indicam de quais ações *intents* um determinado componente deve ser notificado. Usualmente são configurados de modo estático no arquivo de *manifest*, atrelados a componentes como *Activities*, *Broadcast Receivers* e *Services*, fazendo com que estes componentes sejam ativados pelo sistema mediante recebimento de um *intent* pré-configurado (GOOGLE, 2013f).

2.4 BOINC em dispositivos móveis

2.4.1 Estratégias para o Desenvolvimento

O desenvolvimento de mecanismos de computar projetos baseados no BOINC vem sendo estudado há alguns anos, tendo sido utilizadas diversas abordagens para sua implantação. Um das estratégias iniciais foi a reescrita do cliente BOINC em Java (BOINCoid), principal linguagem de programação suportada pelo Android. Esta abordagem foi utilizada por Oded Ben-Dov, que obteve êxito na adaptação do projeto SETI@Home, juntamente com uma implementação simplificada do cliente BOINC, sendo esta abordagem de âmbito conceitual, demonstrando a viabilidade de utilizar dispositivos baseados nessa plataforma. Todavia, tal estratégia apresentou-se inviável em termos práticos, pois exigiria que fossem mantidos dois projetos isolados, cada um com código próprio, o que demandaria excessivo esforço por parte dos desenvolvedores em prol da compatibilidade entre as partes.

Outra estratégia abordada por Peter Hanappe foi o desenvolvimento de um cliente semelhante, chamado BOINCLite (BOINCLITE, 2011), sendo desejada a definição e implementação de uma API simplificada, permitindo acesso a algumas das principais funcionalidades do cliente nativo. Cogitou-se utilizar o BOINCLite como base para o desenvolvimento do cliente Android, sendo a hipótese rejeitada por ser considerada muito restritiva devido ao fato de comportar a participação do voluntário em apenas um projeto, além de impossibilitar a execução simultânea de tarefas.

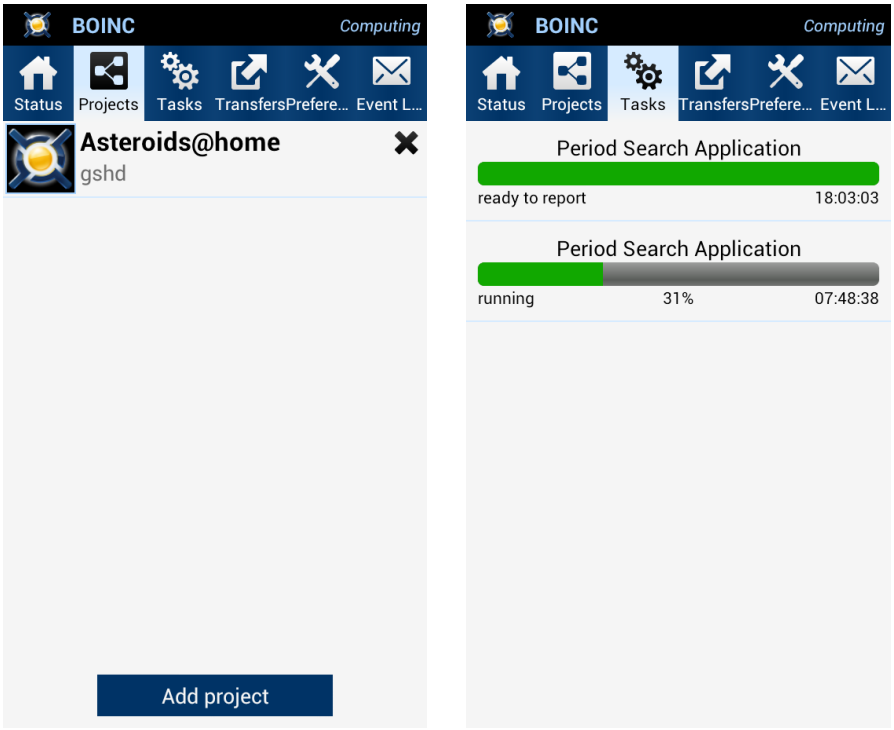
Durante a BOINC Workshop (ANDROBOINC, 2013), realizada no ano de 2011, um grupo de desenvolvedores composto por Keith Uplinger, Peter Hanappe e Michael Tarantido averiguou a hipótese de utilizar-se as principais funcionalidades do cliente em dispositivos baseados na arquitetura de CPU ARM, tendo sido possível concluir que era viável a execução de tais funcionalidades. Um trabalho derivado ocorreu em março de 2012, com uma implementação funcional de um cliente nativo, NativeBOINC, que obteve sucesso em sua implementação mas ainda requeria intervenção direta de administradores do projeto, a fim de adicionar suporte ao Android.

2.4.2 Cliente para Android

Atualmente, existe uma versão funcional do cliente sendo desenvolvida por diversos colaboradores do BOINC, com suporte à maior parte das funcionalidades incluídas na versão convencional do cliente. A aplicação foi obtida através da compilação do cliente existente, com algumas adaptações, a fim de permitir que fossem utilizadas as ferramentas de compilação cruzada disponíveis no Android Native Development Toolkit, gerando um arquivo binário apropriado para a arquitetura ARM e capaz de comunicar-se bilateralmente com a aplicação através de mecanismos de invocação remota de procedimentos (ANDROBOINC, 2013). Algumas das principais vantagens deste modelo de implementação são:

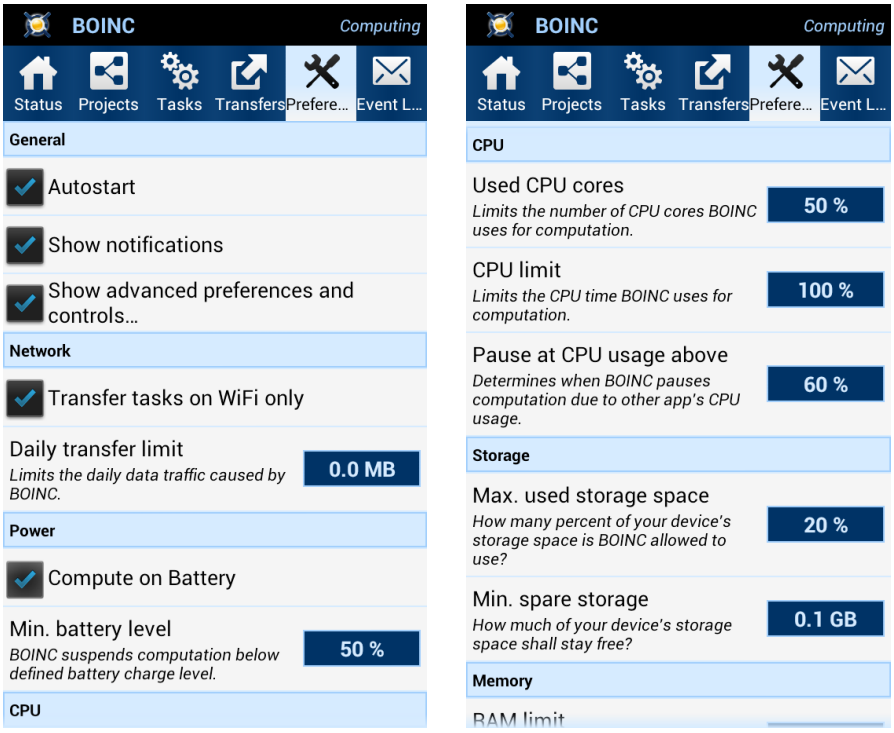
- O desenvolvimento mantém-se centralizado, simplificando a implementação de novas funcionalidades para todas as plataformas suportadas;
- Minimiza o esforço requerido por desenvolvedores, sendo que a maioria dos projetos existentes precisa apenas recompilar seu código utilizando a *toolchain* de compilação cruzada do Android;
- Permite manter a compatibilidade entre os clientes;

O aplicativo, ainda em desenvolvimento, já apresenta algumas das funcionalidades principais, como adicionar projetos à (figura 2.5a) lista e executar tarefas (figura 2.5b). Também é possível configurar diversas opções relativas a celulares (figuras 2.6a e 2.6b), como consumo máximo de memória, armazenamento, utilização do processador, nível mínimo de bateria para execução de tarefas, etc.



(a) Anexação de projetos (b) Execução de Tarefas

Figura 2.5: Funcionalidades suportadas



(a) Opções Diversas - Primeira Parte (b) Opções Diversas - Segunda Parte

Figura 2.6: Configurações do Cliente

3 DESENVOLVIMENTO

A pesquisa realizada por esse trabalho é do âmbito aplicado, tendo sido utilizada uma metodologia experimental. A possibilidade de conclusão de um cliente BOINC voltado para dispositivos móveis foi averiguada através da colaboração no desenvolvimento do software para esta plataforma, incluindo modificações diretas ao código fonte do cliente nativo.

Em fase primária, foi criado um projeto baseado no BOINC, instalado em uma máquina virtual executando Debian, em sua sexta versão. Tal etapa foi necessária para fundamentar conceitos práticos a respeito da plataforma, a fim de permitir que fossem identificadas estratégias mais prováveis para a implementação do cliente. Obteve-se êxito na utilização do projeto, tendo sido possível realizar testes com três máquinas em rede local.

Com base nos conceitos obtidos pela execução da tarefa anterior, foram identificadas características arquiteturais do BOINC, através de estudo do código fonte disponível no repositório do projeto. Nesta etapa, observou-se que havia separação bem definida entre responsabilidades do cliente, do seu gerenciador e da interface gráfica utilizada pelo usuário, fator essencial para possibilitar a execução deste trabalho. Devido à separação de componentes da arquitetura do *BOINC* (figura 2.1), é necessária a implementação de algum meio de comunicação entre os componentes, o que foi realizado por um mecanismo de *Remote Procedure Call (RPC)*, baseado na troca de mensagens codificadas em *XML* (figura 3.1).

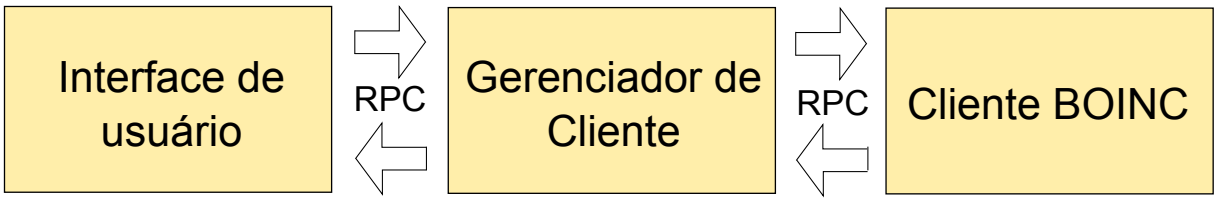


Figura 3.1: Arquitetura do BOINC

3.1 Estratégia Inicial de Implementação

Inicialmente, este trabalho tinha pretensão de realizar uma implementação do cliente utilizando as bibliotecas fornecidas pelo *Software Development Kit* do Android, utilizando *Java Native Interface (JNI)* para realizar chamadas diretas de métodos nativos expostos pela compilação cruzada do cliente nativo, realizada através do Android NDK. Haveria também, nesta arquitetura inicialmente projetada, separação entre o cliente executor de tarefas e a interface

com o usuário (figura 3.2).

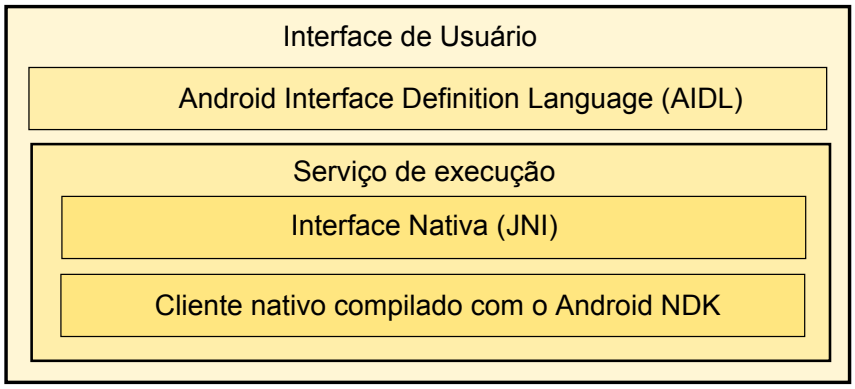


Figura 3.2: Arquitetura Planejada Inicialmente

Esta estratégia foi abandonada após a descoberta e revisão de uma linha de trabalho já existente no repositório central, utilizando arquitetura semelhante e com proposta similar à deste trabalho.

3.2 Análise do Trabalho Existente

Através do estudo do código existente, foi encontrado uma implementação parcial das funcionalidades propostas por esse trabalho, e dada a natureza colaborativa do projeto, foi realizada análise detalhada do trabalho já realizado. Durante essa parte do projeto, foi concluído que a abordagem de isolamento de componentes (cliente, gerenciador e interface), em conjunto com o fato do Android ser baseado em GNU/Linux, permite que a maior parte do código existente seja reaproveitada para o desenvolvimento do cliente voltado para a plataforma alvo deste trabalho.

A instalação do cliente foi relativamente complexa, uma vez que não há pacotes compilados disponíveis, o que requer a configuração de um ambiente de desenvolvimento com uma série de requisitos para a compilação do cliente. Em uma primeira tentativa, foi utilizada uma máquina virtual utilizando como sistema operacional o Arch Linux, em sua variante de 64 bits. Não obteve-se sucesso na compilação cruzada do BOINC nesse sistema, devido a incompatibilidades em uma das bibliotecas de terceiros utilizadas pelo projeto, a libcurl versão 7.27.0.

O ambiente que permitiu a continuidade do processo de instalação pode ser visto na figura 3.3, onde foi utilizada outra máquina virtual com sistema operacional Debian Squeeze 32 bits, utilizando uma *toolchain* para a versão 2.3.3 do Android, criada por intermédio do Android NDK versão R8E.

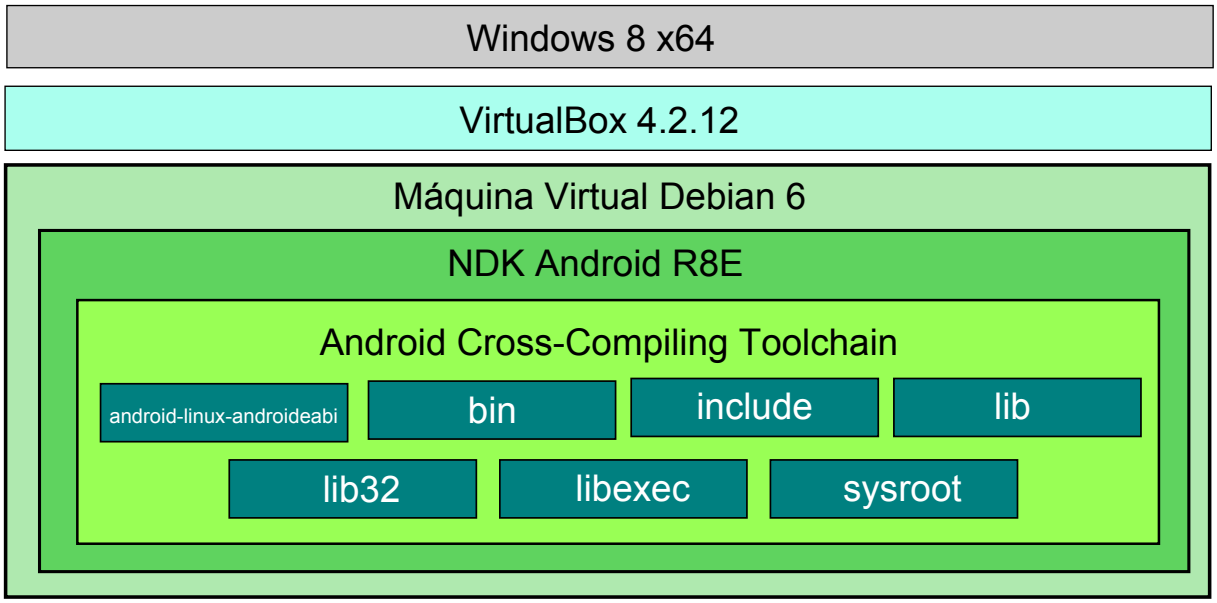


Figura 3.3: Ambiente de Compilação Utilizado

O Android NDK é um pacote de ferramentas, oferecido pela Google Inc., que permite o desenvolvimento de aplicativos utilizáveis no sistema operacional escritos em C/C++, através de interfaces JNI, linguagem na qual o cliente BOINC foi escrito. A *toolchain* gerada inclui compiladores e ligadores capazes de gerar binários nativos para a arquitetura ARM, o que permitiu a geração de dois arquivos binários, *boinc* e *boincmd*, embutidos no pacote de aplicação posteriormente instalado no dispositivo utilizado para os testes.

Após a instalação da aplicação, pôde-se observar que já existe suporte a diversas funcionalidades do BOINC em dispositivos Android, como adição e remoção de projetos, iniciar e pausar tarefas e obter unidades de trabalho. A aplicação também permite o gerenciamento de questões específicas de dispositivos móveis, como consumo de bateria, utilização de processador, memória e armazenamento. Apesar destas funcionalidades existentes, constatou-se, após contato com o grupo de desenvolvedores do sistema, que ainda existem tarefas pendentes que tornem a utilização do aplicativo possível para usuários finais. Como o projeto é de natureza colaborativa, optou-se por auxiliar na conclusão de tarefas restantes.

3.3 Tratamento de Atualizações de Aplicação

3.3.1 Justificativa

Uma das necessidades de um aplicativo como o BOINC é que a aplicação trate eventos de atualização de pacote, pois, ao atualizar uma aplicação, seja de modo automático ou com in-

tervenção do usuário, o Android encerra os processos sendo executados pertencentes ao pacote em sua versão anterior.

No caso do BOINC, este fator atinge sua arquitetura, pois o serviço monitor utilizado para controlar o cliente seria finalizado quando a aplicação fosse atualizada. O principal problema gerado por esta operação é que o processamento por parte do cliente seria parado, o que é especialmente problemático quando considerado que a computação voluntária visa tomar proveito da ociosidade do dispositivo. A interrupção do processamento pode ocasionar que o usuário não perceba que o serviço foi finalizado, não reinicializando a aplicação até que este fato seja percebido.

3.3.2 Requisitos Funcionais

- O serviço monitor deve ser reiniciado após substituição do pacote de aplicação, a fim de retomar atividades anteriores à modificação
- A aplicação deve operar somente sobre o pacote aos qual é autoritária, ou seja, no qual está contida

3.3.3 Implementação

Uma abordagem é a criação de um *intent-filter* para a ação *PACKAGE_REPLACED*, sendo que este *intent* será disparado pelo Android sempre que o pacote da aplicação for atualizado, permitindo que o objeto recebedor execute ações necessárias para retomar as atividades do serviço. Este filtro foi adicionado através da modificação do arquivo *AndroidManifest.xml* (listagem 3.1), responsável pela representação dos dados de manifestação da aplicação no sistema, permitindo que o novo componente fosse detectado pelo Android. O arquivo completo pode ser visto no apêndice A.1.

Listagem de Código 3.1: AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <!--
3    This file is part of BOINC.
4    http://boinc.berkeley.edu
5    Copyright (C) 2012 University of California
6
7    BOINC is free software; you can redistribute it and/or modify it
8    under the terms of the GNU Lesser General Public License
9    as published by the Free Software Foundation,
10   either version 3 of the License, or (at your option) any later version.
11
```

```
12   BOINC is distributed in the hope that it will be useful,
13   but WITHOUT ANY WARRANTY; without even the implied warranty of
14   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
15   See the GNU Lesser General Public License for more details.
16
17   You should have received a copy of the GNU Lesser General Public
         License
18   along with BOINC.  If not, see <http://www.gnu.org/licenses/>.
19 -->
20 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
21     package="edu.berkeley.boinc"
22
23     android:versionCode="17"
24     android:versionName="7.2.0"
25
26     android:installLocation="internalOnly" > <!-- installation on SD card
           would break boot receiver -->
27
28   <!--
29     ...
30   -->
31
32     <!-- Required Permissions -->
33     <uses-permission android:name="android.permission.INTERNET"/>
34     <uses-permission android:name="android.permission.
          RECEIVE_BOOT_COMPLETED"/>
35     <uses-permission android:name="android.permission.WAKE_LOCK"/>
36     <uses-permission android:name="android.permission.
          ACCESS_NETWORK_STATE"/>
37     <uses-permission android:name="android.permission.READ_LOGS"/>
38   <!-- Permissao necessaria para permitir que pacotes sejam reiniciados
         -->
39   <uses-permission android:name="android.permission.RESTART_PACKAGES"/>
40
41     <application
42         android:icon="@drawable/boinc"
43         android:label="@string/app_name"
44         android:theme="@style/Theme"
45         android:allowBackup="true"
46     android:debuggable="true">
47     <activity
48             android:name="edu.berkeley.boinc.BOINCActivity"
49             android:label="@string/app_name">
50             <intent-filter>
51                 <action android:name="android.intent.action.MAIN" />
52                 <category android:name="android.intent.category.LAUNCHER"
                       />
53             </intent-filter>
54     </activity>
55     <!--
56        ...
57     -->
58     <!-- Activity de preferencias -->
59         <ativity android:name=".PrefsActivity" />
60
61     <!--
62        ...
63     -->
```

```
64
65          <receiver android:name=".receiver.BootReceiver">
66              <intent-filter>
67                  <action android:name="android.intent.action.
                        BOOT_COMPLETED"/>
68              </intent-filter>
69      </receiver>
70
71      <!-- Receiver registrado para a acao PACKAGE_REPLACED -->
72          <receiver android:name=".receiver.PackageReplacedReceiver">
73              <intent-filter>
74                  <action android:name="android.intent.action.
                        PACKAGE_REPLACED"/>
75                  <data android:scheme="package" android:path="edu.berkeley
                        .boinc" />
76              </intent-filter>
77          </receiver>
78
79      </application>
80
81  </manifest>
```

Primeiramente, foi necessário adicionar a permissão *RESTART_PACKAGES* (linha 39) na lista de permissões exigidas pela aplicação para a instalação do pacote, o que garante à aplicação privilégios para reiniciar um pacote instalado no sistema, o que, no caso da funcionalidade desejada, corresponde ao pacote da própria aplicação solicitante. Além disso, foram adicionadas as linhas 72 a 77, que correspondem ao registro de uma entidade recebedora do *intent* anteriormente citado, enviado em *broadcast* pelo sistema e recebido apenas por objetos previamente registrados no arquivo *AndroidManifest.xml*.

O recebimento do *intent* contendo a ação reportada é feito através da implementação de uma sub-classe de *BroadcastReceiver*, onde é realizada a sobrescrita do método *onReceive*, que recebe como parâmetros um objeto do tipo *Context* e outro do tipo *Intent*. Na implementação do método, recupera-se os dados do pacote que foi substituído do método *getDataString* de *Intent*, utilizando-o para verificar se a cadeia de caracteres retornada contém o nome do pacote do projeto, e, caso isto seja verdadeiro, dispara-se um novo *intent* para iniciar o serviço monitor, responsável pelo gerenciamento do cliente BOINC no Android. O código completo desenvolvido pode ser visto no apêndice A.3

Listagem de Código 3.2: PackageReplacedReceiver.java

```
1  /*****************************************************************
2   * This file is part of BOINC.
3   * http://boinc.berkeley.edu
4   * Copyright (C) 2012 University of California
5   *
6   * BOINC is free software; you can redistribute it and/or modify it
7   * under the terms of the GNU Lesser General Public License
```

```
 8  * as published by the Free Software Foundation,
 9  * either version 3 of the License, or (at your option) any later version
       .
10  *
11  * BOINC is distributed in the hope that it will be useful,
12  * but WITHOUT ANY WARRANTY; without even the implied warranty of
13  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
14  * See the GNU Lesser General Public License for more details.
15  *
16  * You should have received a copy of the GNU Lesser General Public
       License
17  * along with BOINC.  If not, see <http://www.gnu.org/licenses/>.
18  ******************************************************************/
19 package edu.berkeley.boinc.receiver;
20
21 import edu.berkeley.boinc.utils.*;
22
23 import edu.berkeley.boinc.client.Monitor;
24 import android.content.BroadcastReceiver;
25 import android.content.Context;
26 import android.content.Intent;
27 import android.util.Log;
28
29 public class PackageReplacedReceiver extends BroadcastReceiver {
30     @Override
31     public void onReceive(Context context, Intent intent) {
32
33       if (intent.getDataString().contains("edu.berkeley.boinc")){
34         Log.d(Logging.TAG,"PackageReplacedReceiver: starting service...")
              ;
35           Intent startServiceIntent = new Intent(context, Monitor.class);
36           context.startService(startServiceIntent);
37       } else Log.d(Logging.TAG,"PackageReplacedReceiver: other package: "
              + intent.getDataString());
38     }
39 }
```

3.4 Suporte ao Armazenamento de Dados em Disco Removível

3.4.1 Justificativa

Ao registrar um dispositivo como colaborador de um projeto de computação voluntária baseado no BOINC, o cliente do sistema encarrega-se de adquirir e armazenar em disco os dados requeridos para a execução de tarefas relacionadas ao projeto, além de realizar o armazenamento de resultados parciais de processamento para posterior envio ao servidor principal do projeto.

Considerando que cada projeto anexado a um cliente irá armazenar seus dados de forma independente aos outros projetos, o acúmulo de informações massivas, como arquivos de entrada e de resultado, pode acabar levando à utilização excessiva de espaço de disco, limitando a

participação de projetos de acordo com o espaço de armazenamento disponível no dispositivo.

Apesar do avanço das tecnologias de hardware disponíveis para *smartphones*, ainda não foi possível atingir os patamares de armazenamento disponíveis em computadores pessoais. A solução empregada em grande parte destes dispositivos é a segmentação de espaços de armazenamento, onde utiliza-se uma memória interna acoplada ao dispositivo, em conjunto com um mecanismo de acesso a discos externos, destacando-se em popularidade os *SD Cards*, o que permite expandir as capacidades de gravação de dados nestes dispositivos.

O Android oferece suporte a este tipo de segmentação, a fim de permitir que aplicativos que necessitem de grande quantidade de informações possam ser instalados em um mesmo sistema, através da expansão de armazenamento permitida pelos *SD Cards*. Diversas aplicações tomam proveito desta tecnologia, permitindo que a memória interna dos dispositivos seja preservada para a instalação de outras aplicações.

No caso do BOINC para Android, ainda não há suporte para utilização de armazenamento externo, o que pode representar uma barreira para voluntários que possuam dispositivos com memória interna reduzida. Além disso a maior parte dos dados podem ser armazenados em disco sem grandes prejuízos de desempenho. Assim, frente às possíveis limitações de armazenamento dos dispositivos Android, é desejável que o BOINC suporte a utilização de cartões de memória externos para gravação de dados de projetos, cuja tarefa este trabalho propõe-se a desenvolver.

3.4.2 Requisitos Funcionais

Para a implementação completo ao armazenamento externo, foram identificados como requisitos funcionais:

- Permitir que o usuário mova dados de projeto para o cartão de memória externo
- Oferecer mecanismos que permitam a configuração desta funcionalidade
- Garantir que os dados armazenados sejam movidos em sua totalidade
- A execução de tarefas deve manter suas características funcionais
- As tarefas a serem executadas devem ser interrompidas no início do processo que move os dados e retomadas após a conclusão do procedimento

- As configurações realizadas devem ser mantidas em todas as sessões da aplicação, salvo problemas causados por elementos terceiros
- Nenhuma das funcionalidades anteriores deve ser afetada de maneira negativa

3.4.3 Implementação de um Sistema para Configuração de Armazenamento em Cartão

A primeira etapa para a implementação do suporte ao armazenamento externo consistiu no desenvolvimento de uma funcionalidade que permitisse ao usuário ativar ou não a nova função. Esta nova funcionalidade foi implementada através da reutilização do sistema de preferências utilizado pelo BOINC, bem como modificações na camada de aplicação desenvolvida para Android.

3.4.3.1 Arquivo de Armazenamento de Preferências

A configuração do *BOINC* é armazenada em um arquivo *XML* chamado *global_preferences_override.xml*. Este, contém um série de *tags* que representam cada um dos atributos configuráveis por usuário do software, sendo este incluído no pacote do *BOINC* instalado pelo usuário.

Um dos requisitos para a implementação na nova preferência de usuário é que os dados configurados permaneçam consistentes em todas as sessões da aplicação, até que o usuário intervenha novamente e altere a opção. Para isso, foi necessário acrescentar uma nova propriedade ao arquivo de configuração, chamada *store_projects_external_disk* (listagem 3.3), o que permitiria que a nova opção fosse armazenada através da reutilização de mecanismos existentes no projeto.

Foi considerada a hipótese de armazenar a opção no Android através das funcionalidades oferecidas pela classe *SharedPreferences*, mas esta opção foi descartada devido a alguns fatores principais:

- Insere dependências de sistema operacional, uma vez que a classe estará disponível apenas em dispositivos que utilizem o Android, o que dificulta a reutilização da funcionalidade em outros dispositivos;
- Fere filosofias arquiteturais do *BOINC*, uma vez que as configurações não permanecerão isoladas da camada do cliente;

- Requer a implementação de um mecanismo específico para comunicação entre o cliente e a interface Android

Listagem de Código 3.3: global_prefs_override.xml

```
<global_preferences>
  <run_on_batteries>0</run_on_batteries>
  <battery_charge_min_pct>90</battery_charge_min_pct>
  <battery_max_temperature>40</battery_max_temperature>
  <run_gpu_if_user_active>0</run_gpu_if_user_active>
  <run_if_user_active>1</run_if_user_active>
  <idle_time_to_run>3.0</idle_time_to_run>
  <suspend_cpu_usage>50.0</suspend_cpu_usage>
  <start_hour>0.0</start_hour>
  <end_hour>0.0</end_hour>
  <net_start_hour>0.0</net_start_hour>
  <net_end_hour>0.0</net_end_hour>
  <max_ncpus_pct>50.0</max_ncpus_pct>
  <leave_apps_in_memory>0</leave_apps_in_memory>
  <dont_verify_images>0</dont_verify_images>
  <work_buf_min_days>0.1</work_buf_min_days>
  <work_buf_additional_days>0.5</work_buf_additional_days>
  <disk_interval>60.0</disk_interval>
  <cpu_scheduling_period_minutes>60.0</cpu_scheduling_period_minutes>
  <disk_max_used_gb>1000.0</disk_max_used_gb>
  <disk_max_used_pct>90.0</disk_max_used_pct>
  <disk_min_free_gb>0.1</disk_min_free_gb>
  <ram_max_used_busy_pct>50.0</ram_max_used_busy_pct>
  <ram_max_used_idle_pct>50.0</ram_max_used_idle_pct>
  <max_bytes_sec_up>0.0</max_bytes_sec_up>
  <max_bytes_sec_down>0.0</max_bytes_sec_down>
  <cpu_usage_limit>100.0</cpu_usage_limit>
  <daily_xfer_limit_mb>0.0</daily_xfer_limit_mb>
  <daily_xfer_period_days>0</daily_xfer_period_days>
  <network_wifi_only>1</network_wifi_only>
  <store_projects_external_disk>0</store_projects_external_disk>
</global_preferences>
```

3.4.3.2 Estruturas GLOBAL_PREFS e GLOBAL_PREFS_MASK

As opções escritas no arquivo de preferências são interpretadas pela classe *GLOBAL_PREFS* (figura 3.4), sendo que esta irá percorrer o arquivo todo com auxílio da classe *XML_PARSERS*, a fim de traduzir a interpretação textual dos dados para interpretações acessíveis diretamente pelos mecanismos da linguagem de programação na qual esta parte da aplicação é escrita, C++. Cada um dos dados obtidos a partir do *XML* será armazenado em atributos da estrutura *GLOBAL_PREFS*.

Além da representação das preferências, a estrutura declara alguns métodos, destacando-se:

1. parse_override: realiza interpretação de um arquivo XML através de um objeto *XML_PARSER*, armazenando os valores de acordo com a máscara enviada pelo contexto invocador

2. write: escreve todos os atributos de preferências para uma representação de arquivo *MIOFILE*, no caso usual, o próprio arquivo de preferências

3. write_subset: funcionamento semelhante ao método *write*, porém escreve apenas os valores que foram marcados na instância da classe *GLOBAL_PREFS_MASK* enviada como argumento

Essa estrutura precisou ser modificada, a fim de permitir que o novo valor inserido no arquivo *XML* seja representado na estrutura. Isso foi obtido através da adição de um atributo do tipo *bool* chamado *store_projects_external_disk* (figura 3.4).

Um estrutura auxiliar utilizada é chamada *GLOBAL_PREFS_MASK* (figura 3.4), que tem como finalidade a representação de sub-grupos de propriedades que serão utilizados por métodos da estrutura *GLOBAL_PREFS*. Foi necessário incluir também um novo atributo nessa estrutura, uma vez que a classe deve possibilitar que a nova configuração seja incluída ou não em grupos selecionáveis.

Ambas as modificações são necessárias, também devido ao *BOINC* utilizar um mecanismo de *Remote Procedure Call* para permitir o acesso das configurações aos diferentes componentes do *software*. As duas estruturas citadas são utilizadas como fonte dos dados que serão enviados em mensagens de resposta referentes à consulta de opções de usuários.

Para possibilitar que a nova preferência fosse acessível em ambas as partes da aplicação, i.e. cliente nativo e interface de usuário, foram necessárias modificações nos métodos *write*, *write_subset* e *parse_override* (Apêndice A.4). A modificação é necessária pois permite que a informação armazenada no novo atributo seja lida e escrita no arquivo *XML*, o que garante a persistência das configurações realizadas, bem como a concordância entre o dado armazenado no atributo e sua representação no arquivo.

O método *write*, responsável pela escrita no arquivo *global_preferences_override.xml* (Apêndice A.2), precisou ser modificado para comportar o novo atributo inserido. A modificação consistiu de uma *tag XML* a lista de *tags* a serem escritas no arquivo de saída, de forma que esta necessita ser homônima à propriedade anteriormente inserida no arquivo padrão de preferências, i.e, *store_projects_external_disk*. Para realizar a escrita, utilizou-se a função

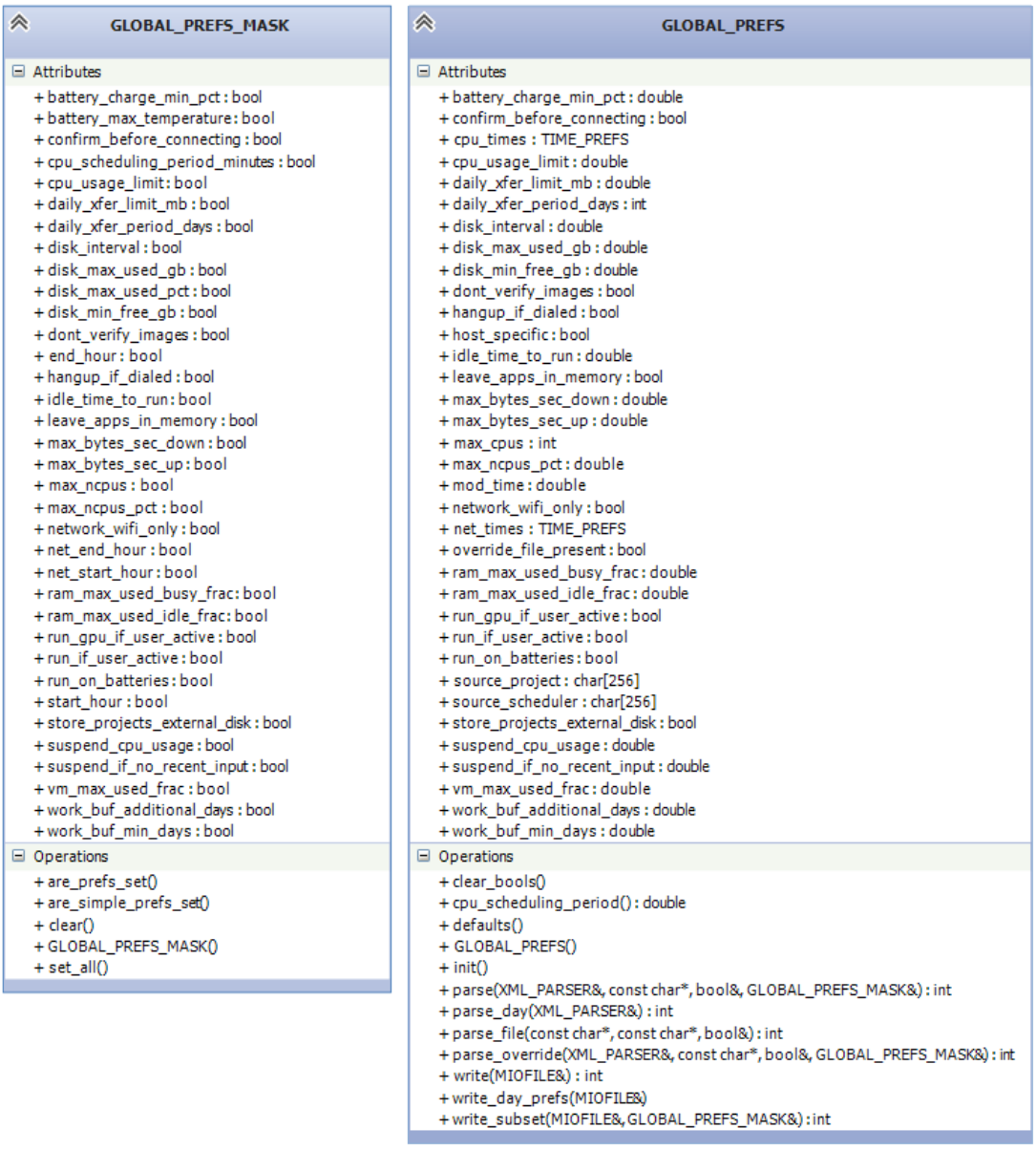


Figura 3.4: Diagrama de Métodos e Atributos das Estruturas *GLOBAL_PREFS_MASK* e *GLOBAL_PREFS*

printf implementada pela classe *MIOFILE*, consistindo em uma implementação semelhante à função nativa da linguagem C, porém, com saída direcionada para o arquivo representada pela instância da classe, ao invés de *stdout*. Sendo assim, a propriedade foi armazenada como uma representação inteira do valor *booleano* do atributo *store_projects_external_disk* da classe *GLOBAL_PREFS* (listagem 3.4).

Listagem de Código 3.4: Modificações realizadas no método write de GLOBAL_PREFS

```
1  int GLOBAL_PREFS::write(MIOFILE& f) {
2      f.printf(
3          "<global_preferences>\n"
4          "    <source_project>%s</source_project>\n"
5          "    <mod_time>%f</mod_time>\n"
6      /*
7        ...
8        ...
9      */
10     // atributo abaixo foi adicionado para permitir o cliente escrevesse
           a propriedade diretamente no arquivo
11         "    <store_projects_external_disk>%d</
               store_projects_external_disk>\n",
12         source_project,
13         mod_time,
14         /*
15       ...
16       ...
17     */
18     //armazena uma representacao inteira do atributo bina'rio da
           estrutura GLOBAL_PREFS
19     store_projects_external_disk ? 1 : 0
20     );
21   /*
22     ...
23     ...
24   */
25     f.printf("</global_preferences>\n");
26     return 0;
27 }
```

Foi necessário modificar o método *write_subset*, responsável pela escrita seletiva de dados de configuração, através da utilização de uma máscara de valores booleanos, representada pela classe *GLOBAL_PREFS_MASK*. Desta forma, utilizou-se mecanismo semelhante ao implementado no método *write*, porém sendo a escrita realizada somente se o atributo da classe *GLOBAL_PREFS_MASK* estiver com valor verdadeiro (listagem 3.5.

Listagem de Código 3.5: Modificações realizadas no método write_subset de GLOBAL_PREFS

```
1  int GLOBAL_PREFS::write_subset(MIOFILE& f, GLOBAL_PREFS_MASK& mask) {
2      if (!mask.are_prefs_set()) return 0;
3
```

```
4       f.printf("<global_preferences>\n");
5       // ...
6     if (mask.store_projects_external_disk) {
7       f.printf("    <store_projects_external_disk>%d</
            store_projects_external_disk>\n", store_projects_external_disk ?
            1:0 );
8     }
9     // ...
10      write_day_prefs(f);
11      f.printf("</global_preferences>\n");
12      return 0;
13  }
```

Por fim, foi necessário adaptar o método *parse_override*, responsável pela interpretação dos dados armazenados no arquivo e sobrescrita dos valores armazenados pela classe *GLOBAL_PREFS*. Para isso, incluiu-se uma nova condição que irá tratar a *tag* inserida, utilizando-se o método *parse_bool* da classe auxiliar *XML_PARSER*, que irá atribuir o valor interpretado do arquivo à referência de uma variável do tipo *bool*, no caso, o próprio atributo *store_projects_external_disk* (listagem 3.6).

Listagem de Código 3.6: Modificações realizadas no método parse_override de GLOBAL_PREFS

```
1  int GLOBAL_PREFS::parse_override(
2      XML_PARSER& xp, const char* host_venue, bool& found_venue,
           GLOBAL_PREFS_MASK& mask
3  ) {
4      /*
5      ...
6    */
7
8      while (!xp.get_tag(attrs, sizeof(attrs))) {
9          if (!xp.is_tag) continue;
10         if (xp.match_tag("global_preferences")) continue;
11         if (xp.match_tag("/global_preferences")) {
12             return 0;
13         }
14     /*
15       ...
16     */
17     //verifica se o elemento atual e' do tipo bool, e se verdadeiro,
           armazena o valor interpretado em store_projects_external_disk
18     if (xp.parse_bool("store_projects_external_disk",
           store_projects_external_disk)) {
19       continue;
20     }
21         xp.skip_unexpected(false, "GLOBAL_PREFS::parse_override");
22     }
23     return ERR_XML_PARSE;
24 }
```

3.4.4 Activity PrefsActivity e Classe RpcClient

A partir das modificações realizadas no cliente para comportar a nova opção de usuário, é necessário que esta seja apresentada ao utilizador através de um interface apropriada. Para isso, foi preciso realizar modificações na classe *PrefsActivity* (Apêndice A.5), responsável pela seção de configuração da aplicação do *BOINC* para Android.

A *activity* utiliza comandos de *RPC* para a obtenção e atualização de preferências estabelecidas pelo usuário. Esta comunicação é realizada com a troca de mensagens codificadas em *XML*, enviadas através de um *socket* de comunicação, utilizado como rota entre cliente e servidor *RPC*. Um dos principais componentes utilizados por esta aplicação é chamado *set_global_prefs_override*, sendo utilizado através da invocação do método *setGlobalPreferences* da classe *Monitor*, com a passagem de um objeto do tipo *GlobalPreferences*. Foram necessárias algumas modificações no código da classe *PrefsActivity* para que pudesse oferecer ao usuário um campo interativo para configuração da nova funcionalidade, resultado no código da listagem 3.7)

Listagem de Código 3.7: Modificações realizadas na classe PrefsActivity

```
1  /*******************************************************************
2   * This file is part of BOINC.
3   * http://boinc.berkeley.edu
4   * Copyright (C) 2012 University of California
5   *
6   * BOINC is free software; you can redistribute it and/or modify it
7   * under the terms of the GNU Lesser General Public License
8   * as published by the Free Software Foundation,
9   * either version 3 of the License, or (at your option) any later version
        .
10  *
11  * BOINC is distributed in the hope that it will be useful,
12  * but WITHOUT ANY WARRANTY; without even the implied warranty of
13  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
14  * See the GNU Lesser General Public License for more details.
15  *
16  * You should have received a copy of the GNU Lesser General Public
        License
17  * along with BOINC.  If not, see <http://www.gnu.org/licenses/>.
18  ******************************************************************/
19
20 /* ... */
21 public class PrefsActivity extends FragmentActivity {
22   /* ... */
23   private void populateLayout() {
24     /* ... */
25     if (ExternalStorageUtil.isExternalStorageWritable() || advanced)
26       data.add(new PrefsListItemWrapper(this,R.string.
             prefs_category_storage,true));
27     if (ExternalStorageUtil.isExternalStorageWritable())
```

```
28        data.add(new PrefsListItemWrapperBool(this, R.string.
              prefs_store_projects_external_disk, R.string.
              prefs_category_storage,clientPrefs.store_projects_external_disk
              ));
29      /* ... */
30    }
31    /* ... */
32    public void onCbClick (View view) {
33      if(Logging.DEBUG) Log.d(Logging.TAG,"onCbClick");
34      Integer ID = (Integer) view.getTag();
35      CheckBox source = (CheckBox) view;
36      Boolean isSet = source.isChecked();
37      switch (ID) {
38      /* ... */
39          case R.string.prefs_store_projects_external_disk:
40              clientPrefs.store_projects_external_disk = isSet;
41              updateBoolPref(ID, isSet);
42              new WriteClientPrefsAsync().execute(clientPrefs);
43              break;
44      }
45    }
46    private void writeClientValuePreference(int id, double value) {
47      switch (id) {
48      /* ... */
49      case R.string.prefs_store_projects_external_disk:
50        clientPrefs.store_projects_external_disk = value;
51        break;
52      /* ... */
53      }
54      updateValuePref(id, value);
55      new WriteClientPrefsAsync().execute(clientPrefs);
56    }
57    /* ... */
58  }
```

O método anteriormente citado é executado no *service Monitor*, delegando o processamento dos comandos *RPC* à classe *RpcClient*. Em seu fluxo de execução, este dispara o comando *setGlobalPrefsOverrideStruct* a um atributo do tipo RpcClient, sendo este comando responsável pela informação das novas preferências ao cliente nativo. Ao término da operação, são requisitadas as preferências atualizadas ao cliente, o que é feito através da invocação do método *readGlobalPrefsOverride*. Por fim é realizada uma chamada à *getGlobalPrefsWorkingStruct*, também definido por *RpcClient*, a fim de obter as preferências que devem ser utilizadas para trabalho, alterando o estado do cliente através de manipulação do objeto do tipo *ClientStatus* obtido pela chamada local ao método *getClientStatus*.

Devido à comunicação entre cliente e interface ser bi-direcional, também foi necessário realizar alterações na parte a aplicação escrita em *Java*, isto é, aquela desenvolvida especificamente para o Android. Primeiramente, foi adicionado um atributo *booleano* público à classe *GlobalPreferences*, responsável pelo armazenamento da opção de usuário correspondente à uti-

lização do cartão *SD* para dados de projeto. Em seguida, foi necessário adicionar uma nova propriedade na requisição *RPC* construída pelo método *setGlobalPrefsOverrideStruct* da classe *RpcClient*, possibilitando que a nova propriedade seja informada ao cliente nativo (listagem 3.8).

Listagem de Código 3.8: Modificações realizadas na classe RpcClient

```
1 /*********************************************************************
2  * This file is part of BOINC.
3  * http://boinc.berkeley.edu
4  * Copyright (C) 2012 University of California
5  *
6  * BOINC is free software; you can redistribute it and/or modify it
7  * under the terms of the GNU Lesser General Public License
8  * as published by the Free Software Foundation,
9  * either version 3 of the License, or (at your option) any later version
       .
10 *
11 * BOINC is distributed in the hope that it will be useful,
12 * but WITHOUT ANY WARRANTY; without even the implied warranty of
13 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
14 * See the GNU Lesser General Public License for more details.
15 *
16 * You should have received a copy of the GNU Lesser General Public
       License
17 * along with BOINC.  If not, see <http://www.gnu.org/licenses/>.
18 *********************************************************************/
19 public class RpcClient {
20   // ...
21   public synchronized boolean setGlobalPrefsOverrideStruct(
         GlobalPreferences globalPrefs) {
22     try {
23       //...
24             mRequest.append("<store_projects_external_disk>");
25             mRequest.append(globalPrefs.store_projects_external_disk ? 1
                   : 0);
26             mRequest.append("</store_projects_external_disk>\n");
27       //...
28       sendRequest(mRequest.toString());
29       receiveReply();
30       return true;
31     } catch (IOException e) {
32       if(Logging.WARNING) Log.w(Logging.TAG, "error in
             setGlobalPrefsOverrideStruct()", e);
33       return false;
34     }
35   }
36 }
```

4 RESULTADOS E DISCUSSÃO

Este capítulo irá apresentar os resultados obtidos com a realização deste trabalho. Nesta etapa, buscou-se apresentar testes e modelos de fluxo para demonstrar o funcionamento das contribuições desenvolvidas e resultados associados. Foram utilizados dois dispositivos para os testes. Primeiro, um *Android Virtual Device*, i.e., um dispositivo virtual capaz de simular o comportamento de um aparelho real criado através da ferramenta *AVD Manager*, incluída no *SDK* do Android (GOOGLE, 2013g). Este foi configurado para utilizar a versão *2.3.3* do Android, com 256 Mb de memória *RAM* e 64 Mb de espaço em armazenamento interno.

Para controle, foi utilizado também um dispositivo físico, sendo este um *smartphone* modelo *Sony Xperia P L22i*, com uma instalação do Android em sua versão 4.1.2. O aparelho utiliza um processador baseado na arquitetura *ARMv7*, possuindo 1 Gb de memória RAM e 16 Gb de armazenamento interno.

4.1 Tratamento de Atualizações de Aplicação

A implementação desta funcionalidade, proposta por este trabalho, foi concluída com êxito. Pôde-se concluir a respeito do funcionamento dos mecanismo desenvolvidos através da observação do comportamento apresentado nos dispositivos instalados.

Como teste, a versão modificada de aplicação foi instalada nos dois dispositivos destinados à averiguação do funcionamento. Isto foi feito através de uma ferramenta incluída no *SDK* do Android, chamada *Android Debug Bridge (ADB)* (GOOGLE, 2013h), intermediada pela interface gráfica do *Integrated Development Environment(IDE)* utilizado para o desenvolvimento, *Android Studio* versão *AI-130.737825* (GOOGLE, 2013i). Após instalação do pacote de aplicação, observou-se que a mesma não foi comprometida pela implementação da nova funcionalidade.

Uma vez observado que a aplicação permanecia estável, utilizou-se o comando "*adb install -r BOINC.apk*" (figura 4.1), que irá realizar a instalação do pacote de aplicação *BOINC.apk* no dispositivo conectado ao *ADB*. A execução deste comando, causa a sobrescrita do pacote anteriormente instalado, fazendo com que a aplicação encerrasse todos seus processos, e consequentemente parasse a execução de tarefas no cliente. Após implementação de um objeto que tratasse o evento relacionado de atualização de pacote pôde-se observar que a interrupção ocorre de fato, porém, graças ao *BroadcastReceiver* desenvolvido, é reiniciada após o término

da operação de atualização, retomando o processamento interrompido.

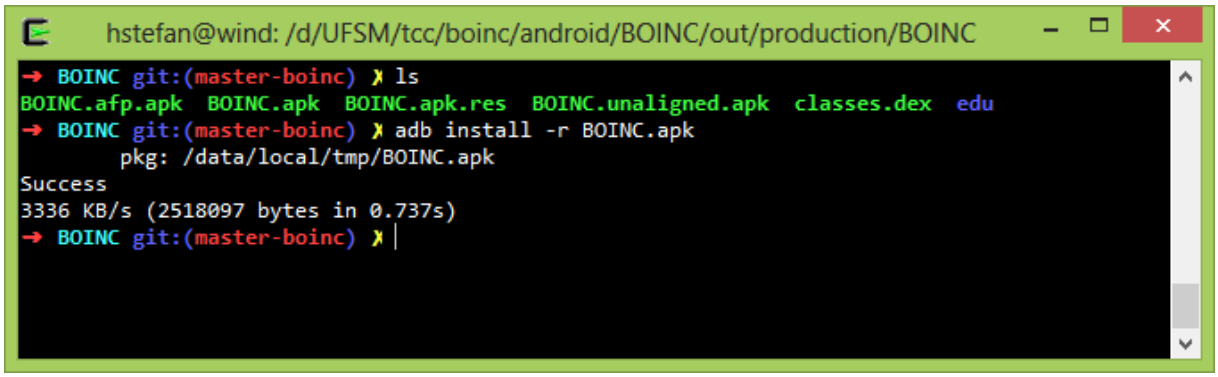


Figura 4.1: Instalação do Pacote de Aplicação em um Dispositivo Android

Após o teste do código desenvolvido, foi gerado um *patch*, através do comando "*git format-patch origin/master --stdout » package-replaced-android.patch*", que, uma vez executado, irá criar um arquivo com a representação de todas as modificações ocorridas entre a revisão informada e o estado atual do repositório (CHACON, 2009). Este tipo de operação é comum em projetos colaborativos que utilizem sistemas de versionamento; no caso do *BOINC*, o controlador de versão utilizado é o *git*, de forma que o *patch* que representa as modificações a serem enviadas pode ser exportado através do comando anterior. Desta forma, o arquivo gerado foi enviado via e-mail diretamente à equipe mantenedora do repositório principal do projeto, de modo que a colaboração pudesse ser incorporada ao projeto. O *patch* enviado foi aceito e integrado ao repositório no dia 7 de Julho de 2013 [1], tendo sido a tarefa removida da lista oficial de funcionalidades restantes (FRITZSCH; ANDERSON; WALTON, 2013).

4.2 Implementação de uma Preferência de Usuário para Utilização de Armazenamento Externo

A tarefa de implementação da preferência que determina se o usuário deseja armazenar os dados de projeto, tinha como objetivo colaborar com o desenvolvimento de outra funcionalidade para aplicação. Devido a questões específicas de dispositivos móveis, é desejável possibilitar que o usuário gerencie suas alternativas de armazenamento; desta forma, a tarefa desenvolvida almeja servir como base para a implementação da nova preferência.

A verificação da funcionalidade implementada foi realizada através da análise do fluxo de execução das partes envolvidas (figura 4.2), identificado através da observação da sequência

[1] O repositório está disponível em `git://boinc.berkeley.edu/boinc-v2.git`, tendo sido a mudança incorporada pelo commit 18b6f26adbb35aa9f9348d16a6fb911f681d701a.

algorítmica definida pelo código fonte do cliente BOINC. Esta abordagem permite que o comportamento da aplicação seja previsto, permitindo que sejam comparados o fluxo desejado e o gerado *de facto*, à fim de atestar corretude dos algoritmos desenvolvidos.

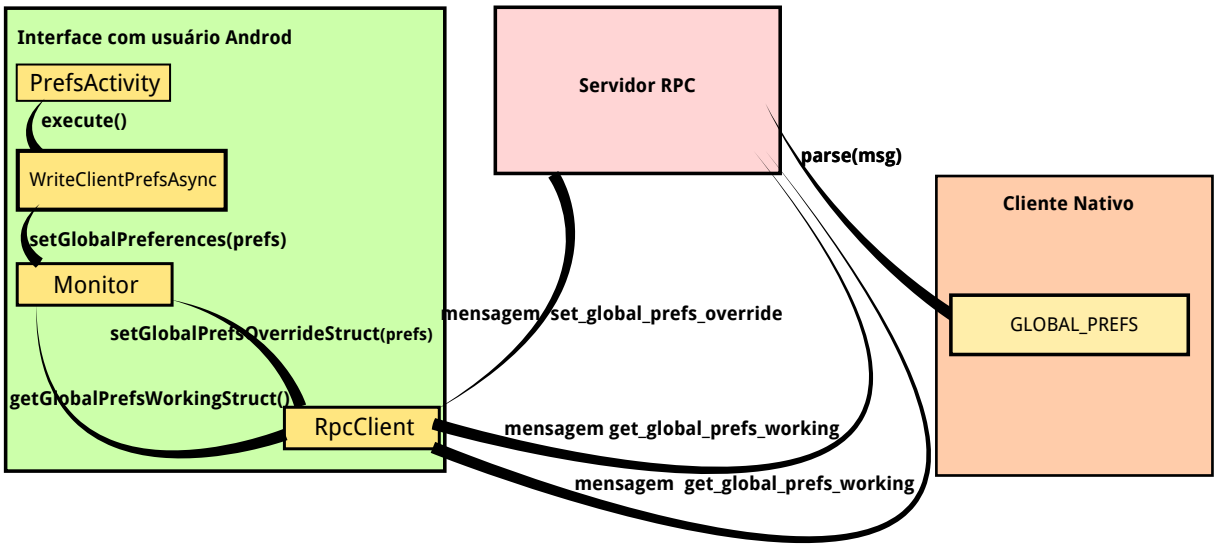


Figura 4.2: Fluxo de Execução das Partes Envolvidas

Ao alterar algum valor da representação das preferências na classe *GlobalPrefs*, será executada uma ação assíncrona responsável pela escrita dos dados no cliente nativo, através da invocação do método *setGlobalPreferences* da classe *Monitor*. Este, por sua vez, irá enviar uma requisição ao servidor *RPC*, através de uma mensagem do tipo *set_global_prefs_override*. O recebimento da mensagem pelo servidor irá acarretar no armazenamento do valor do atributo no arquivo de configurações, uma vez que a classe alvo (GLOBAL_PREFS) foi modificada para comportar o novo atributo. Após o retorno da função *setGlobalPreferences*, o objeto *Monitor* irá invocar o método *getGlobalPrefsWorkingStruct*, que enviará de uma mensagem do tipo *get_global_prefs_working* ao servidor *RPC*, que irá obter o resultado através do comando *parse* da classe *GLOBAL_PREFS*.

Sendo assim, mediante análise do fluxo de execução da aplicação, foi possível identificar que a nova funcionalidade comporta-se como esperado, pois os requisitos funcionais desta são atendidos. O código desenvolvido não foi encaminhado ao repositório central, pois não soluciona a totalidade da tarefa planejada para o suporte à cartões *SD*. Todavia, algumas estratégias foram discutidas para a implementação completa desta outra funcionalidade, juntamente com a equipe de mantenedores do *BOINC*.

5 CONCLUSÃO E TRABALHO FUTUROS

Com os avanços tecnológicos dos computadores, estes tornaram-se ferramentas de grande utilidade para projetos científicos. Algumas pesquisas podem envolver grandes quantidades de informação, bem como procedimentos de alta complexidade para a análise dos dados, o que pode dificultar o andamento de uma pesquisa, se os procedimentos forem executados somente por humanos. Sendo assim, pode-se utilizar computadores para a execução de tais procedimentos, uma vez que estes se mostram mais eficazes do que seres humanos na realização de longos processamentos.

Porém, os algoritmos utilizados para um projeto científico podem tornar-se complexos até para computadores, levando ao surgimento da necessidade de adquirir recursos mais poderosos. Obter aparelhos apropriados para o processamento nem sempre é uma alternativa viável, pois estes geralmente envolvem altos custos financeiros, o que pode inviabilizar a execução de projetos com orçamento inferior ao necessário.

A computação voluntária é uma ideologia que propõe-se a reduzir os custos envolvidos com a disponibilização de maior poder computacional, uma vez que tem como ideia central viabilizar a doação de processamento por usuários voluntários. Uma das possibilidades mais populares para a utilização desta ideia é implementada pelo *BOINC*, que constitui uma infraestrutura voltada para processamento em grade e para computação voluntária.

O *BOINC* é suportado por diversos sistemas operacionais, porém, ainda não possui suporte ao Android, sistema voltado para dispositivo móveis que torna-se cada vez mais popular. Com o avanço das tecnologias empregadas nestes dispositivos, estes tornam-se uma fonte de possíveis voluntários com grande potencial, porém encontrando-se atualmente desperdiçada devido a ausência de suporte à plataforma. Este trabalho colaborou com o avanço da implementação das funcionalidades necessárias para oferecer suporte ao Android, tendo sido implementado um melhor gerenciamento de atualizações do pacote da aplicação. A funcionalidade desenvolvida foi aprovada pela equipe de desenvolvedores e encontra-se em uso pelo sistema.

Este projeto também iniciou o trabalho necessário para melhor utilizar os recursos de armazenamento de dispositivos móveis, uma vez que estes comumente possuem cartões de memória passiveis de escrita e armazenamento de dados. O trabalho aqui realizado consistiu na adição de uma nova preferência de usuário, a qual pôde ser exposta com sucesso para o cliente executor e a interface de usuário.

O trabalho desenvolvido não conclui a lista de pendências para o lançamento de uma pacote para usuários finais. Para trabalhos futuros, o autor deste trabalho pretende continuar a implementação do suporte a cartões de memória, uma vez que o código desenvolvido não realiza a movimentação dos dados. Também ficam aqui enumeradas como pendentes a implementação de suporte a gerenciadores de conta, dispositivos Android baseados na arquitetura x86, além de atividades menores inseridas diariamente na página de tarefas do projeto.

REFERÊNCIAS

ALLIANCE, O. H. **Android Review**. Acessado em Julho/2013, http://www.openhandsetalliance.com/android_overview.html.

ANDERSON, D. P. **Basic concepts**. Acessado em Junho/2013, http://boinc.berkeley.edu/trac/wiki/BasicConcepts.

ANDERSON, D. P. et al. **SETI@Home**. Acessado em Maio/2013, http://setiathome.berkeley.edu.

ANDERSON, D. P.; MAIRE, N. **Overview of BOINC**. Acessado em Junho/2013, http://boinc.berkeley.edu/trac/wiki/BoincIntro.

ANDROBOINC. **AndroBOINCImpl**. Acessado em Fevereiro/2013, http://boinc.berkeley.edu/trac/wiki/AndroidBoincImpl.

BOINCLITE. **BoincLite**. Acessado em Fevereiro/2013, http://boinc.berkeley.edu/trac/wiki/BoincLite.

CHACON, S. **Pro Git**. 1ª.ed. [S.l.]: Apress, 2009.

EASTLACK, J. R. **Extending Volunteer Computing to Mobile Devices**. 2011. Dissertação (Mestrado em Ciência da Computação) — New Mexico State University.

FRITZSCH, J.; ANDERSON, D. P.; WALTON, R. **Getting Started with Android Studio**. Acessado em Julho/2013, boinc.berkeley.edu/trac/wiki/AndroidBoincTodo.

GOOGLE. **Activities**. Acessado em Julho/2013, http://developer.android.com/guide/components/activities.html.

GOOGLE. **Services**. Acessado em Julho/2013, http://developer.android.com/guide/components/services.html.

GOOGLE. **The AndroidManifest.xml File**. Acessado em Julho/2013, http://developer.android.com/guide/topics/manifest/manifest-intro.html.

GOOGLE. **BroadcastReceiver**. Acessado em Julho/2013, http://developer.android.com/reference/android/content/BroadcastReceiver.html.

GOOGLE. **LocalBroadcastReceiver**. Acessado em Julho/2013, http://developer.android.com/reference/android/support/v4/content/LocalBroadcastManager.html.

GOOGLE. **Intents and Intent Filters**. Acessado em Julho/2013, http://developer.android.com/guide/components/intents-filters.html.

GOOGLE. **Managing AVDs with AVD Manager**. Acessado em Julho/2013, http://developer.android.com/tools/devices/managing-avds.html.

GOOGLE. **Android Debug Bridge**. Acessado em Julho/2013, http://developer.android.com/tools/help/adb.html.

GOOGLE. **Getting Started with Android Studio**. Acessado em Julho/2013, http://developer.android.com/sdk/installing/studio.html.

PITKANEN, P. Comparing Google's Android and Apple's iOS Mobile Software Development Environments. , [S.l.], 2008.

QUALCOMN, T. **Snapdragon™ Mobile Processors3**. Acessado em Junho/2013, https://developer.qualcomm.com/discover/chipsets-and-modems/snapdragon.

RIVERA, J.; MEULEN, R. v. d. **Gartner Says Asia/Pacific Led Worldwide Mobile Phone Sales to Growth in First Quarter of 2013**. Acessado em Março/2013, http://www.gartner.com/newsroom/id/2482816.

SARMENTA, L. F. **Volunteer Computing**. 2001. Dissertação (Mestrado em Ciência da Computação) — Massachusetts Institute of Technology.

WEEK, B. **Google Buys Android for Its Mobile Arsenal**. Acessado em Julho/2013, http://www.businessweek.com/stories/2005-08-16/google-buys-android-for-its-mobile-arsenal.

APÊNDICES

APÊNDICE A – Arquivos Modificados ou Inseridos

A.1 AndroidManifest.xml

Listagem de Código A.1: AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <!--
3    This file is part of BOINC.
4    http://boinc.berkeley.edu
5    Copyright (C) 2012 University of California
6
7    BOINC is free software; you can redistribute it and/or modify it
8    under the terms of the GNU Lesser General Public License
9    as published by the Free Software Foundation,
10   either version 3 of the License, or (at your option) any later version.
11
12   BOINC is distributed in the hope that it will be useful,
13   but WITHOUT ANY WARRANTY; without even the implied warranty of
14   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
15   See the GNU Lesser General Public License for more details.
16
17   You should have received a copy of the GNU Lesser General Public
        License
18   along with BOINC.  If not, see <http://www.gnu.org/licenses/>.
19 -->
20 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
21     package="edu.berkeley.boinc"
22
23     android:versionCode="21"
24     android:versionName="7.2.4"
25
26     android:installLocation="internalOnly" > <!-- installation on SD card
           would break boot receiver -->
27
28     <!-- Add Google Play store metadata informing the store we can run on
           tablets and other large screen devices -->
29     <supports-screens
30         android:anyDensity="true"
31         android:smallScreens="true"
32         android:normalScreens="true"
33         android:largeScreens="true"
34         android:xlargeScreens="true" />
35
36     <!-- CAUTION: increasing targetSDK to >9 removes menu button on new
          devices  -->
37     <uses-sdk
38         android:minSdkVersion="9"
39         android:targetSdkVersion="9" />
40
41     <!-- Required Permissions -->
42     <uses-permission android:name="android.permission.INTERNET"/>
43     <uses-permission android:name="android.permission.
          RECEIVE_BOOT_COMPLETED"/>
44     <uses-permission android:name="android.permission.WAKE_LOCK"/>
```

```
45      <uses-permission android:name="android.permission.
            ACCESS_NETWORK_STATE"/>
46      <uses-permission android:name="android.permission.READ_LOGS"/>
47      <uses-permission android:name="android.permission.RESTART_PACKAGES"/>
48      <uses-permission android:name="android.permission.
            WRITE_EXTERNAL_STORAGE"/>
49
50      <application
51          android:icon="@drawable/boinc"
52          android:label="@string/app_name"
53          android:theme="@style/Theme"
54          android:allowBackup="true"
55          android:debuggable="true"> <!-- android:largeHeap="true" ,
                possible api level > 10. grants application larger dalvik vm
                heap. better performance when showing many slideshow pictures
                . -->
56
57          <activity
58              android:name="edu.berkeley.boinc.BOINCActivity"
59              android:label="@string/app_name">
60              <intent-filter>
61                  <action android:name="android.intent.action.MAIN" />
62                  <category android:name="android.intent.category.LAUNCHER"
                        />
63              </intent-filter>
64          </activity>
65          <activity android:name=".AttachProjectListActivity"/>
66          <activity android:name=".AttachProjectLoginActivity"
67              android:configChanges="orientation"/>
68          <activity android:name=".AttachProjectRegistrationActivity"/>
69          <activity android:name=".AttachProjectWorkingActivity"
70              android:configChanges="orientation"/>
71          <activity android:name=".StatusActivity" />
72          <activity android:name=".TasksActivity" />
73          <activity android:name=".TransActivity" />
74          <activity android:name=".PrefsActivity" />
75          <activity android:name=".ProjectsActivity" />
76          <activity android:name=".EventLogActivity" />
77          <service android:name=".client.Monitor"></service>
78          <receiver android:name=".receiver.BootReceiver">
79              <intent-filter>
80                  <action android:name="android.intent.action.
                        BOOT_COMPLETED"/>
81              </intent-filter>
82          </receiver>
83          <receiver android:name=".receiver.PackageReplacedReceiver">
84              <intent-filter>
85                  <action android:name="android.intent.action.
                        PACKAGE_REPLACED"/>
86                  <data android:scheme="package" android:path="edu.berkeley
                        .boinc" />
87              </intent-filter>
88              <intent-filter>
89                  <action android:name="android.intent.action."/>
90                  <data android:scheme="package" android:path="edu.berkeley
                        .boinc" />
91              </intent-filter>
92          </receiver>
```

```
93
94      </application>
95
96  </manifest>
```

A.2 global_prefs_override.xml

Listagem de Código A.2: global_prefs_override.xml

```
1  <global_preferences>
2    <run_on_batteries>0</run_on_batteries>
3    <battery_charge_min_pct>90</battery_charge_min_pct>
4    <battery_max_temperature>40</battery_max_temperature>
5    <run_gpu_if_user_active>0</run_gpu_if_user_active>
6    <run_if_user_active>1</run_if_user_active>
7    <idle_time_to_run>3.0</idle_time_to_run>
8    <suspend_cpu_usage>50.0</suspend_cpu_usage>
9    <start_hour>0.0</start_hour>
10   <end_hour>0.0</end_hour>
11   <net_start_hour>0.0</net_start_hour>
12   <net_end_hour>0.0</net_end_hour>
13   <max_ncpus_pct>50.0</max_ncpus_pct>
14   <leave_apps_in_memory>0</leave_apps_in_memory>
15   <dont_verify_images>0</dont_verify_images>
16   <work_buf_min_days>0.1</work_buf_min_days>
17   <work_buf_additional_days>0.5</work_buf_additional_days>
18   <disk_interval>60.0</disk_interval>
19   <cpu_scheduling_period_minutes>60.0</cpu_scheduling_period_minutes>
20   <disk_max_used_gb>1000.0</disk_max_used_gb>
21   <disk_max_used_pct>90.0</disk_max_used_pct>
22   <disk_min_free_gb>0.1</disk_min_free_gb>
23   <ram_max_used_busy_pct>50.0</ram_max_used_busy_pct>
24   <ram_max_used_idle_pct>50.0</ram_max_used_idle_pct>
25   <max_bytes_sec_up>0.0</max_bytes_sec_up>
26   <max_bytes_sec_down>0.0</max_bytes_sec_down>
27   <cpu_usage_limit>100.0</cpu_usage_limit>
28   <daily_xfer_limit_mb>0.0</daily_xfer_limit_mb>
29   <daily_xfer_period_days>0</daily_xfer_period_days>
30   <network_wifi_only>1</network_wifi_only>
31   <store_projects_external_disk>0</store_projects_external_disk>
32 </global_preferences>
```

A.3 PackageReplacedReceiver.java

Listagem de Código A.3: PackageReplacedReceiver.java

```
1  /*****************************************************************
2   * This file is part of BOINC.
3   * http://boinc.berkeley.edu
4   * Copyright (C) 2012 University of California
5   *
6   * BOINC is free software; you can redistribute it and/or modify it
7   * under the terms of the GNU Lesser General Public License
8   * as published by the Free Software Foundation,
```

```
 * either version 3 of the License, or (at your option) any later version.
 *
 * BOINC is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with BOINC.  If not, see <http://www.gnu.org/licenses/>.
 ******************************************************************/
package edu.berkeley.boinc.receiver;

import edu.berkeley.boinc.utils.*;

import edu.berkeley.boinc.client.Monitor;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;

public class PackageReplacedReceiver extends BroadcastReceiver {

  /*
   * Receiver for android.intent.action.PACKAGE_REPLACED
   * This intent is protected and can only be triggered by the system
   * To test this code run ADB with the following command:
   * adb install -r yourapp.apk
   */
    @Override
    public void onReceive(Context context, Intent intent) {

      if (intent.getDataString().contains("edu.berkeley.boinc")){
        Log.d(Logging.TAG,"PackageReplacedReceiver: starting service...");
          Intent startServiceIntent = new Intent(context, Monitor.class);
          context.startService(startServiceIntent);
      } else Log.d(Logging.TAG,"PackageReplacedReceiver: other package: "
          + intent.getDataString());
    }
}
```

A.4 prefs.cpp

Listagem de Código A.4: prefs.cpp

```
// This file is part of BOINC.
// http://boinc.berkeley.edu
// Copyright (C) 2008 University of California
//
// BOINC is free software; you can redistribute it and/or modify it
// under the terms of the GNU Lesser General Public License
// as published by the Free Software Foundation,
// either version 3 of the License, or (at your option) any later version.
//
```

```
10 // BOINC is distributed in the hope that it will be useful,
11 // but WITHOUT ANY WARRANTY; without even the implied warranty of
12 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
13 // See the GNU Lesser General Public License for more details.
14 //
15 // You should have received a copy of the GNU Lesser General Public
       License
16 // along with BOINC.  If not, see <http://www.gnu.org/licenses/>.
17
18 #if   defined(_WIN32) && !defined(__STDWX_H__)
19 #include "boinc_win.h"
20 #elif defined(_WIN32) && defined(__STDWX_H__)
21 #include "stdwx.h"
22 #else
23 #include "config.h"
24 #include <cstdio>
25 #include <cstring>
26 #include <cstdlib>
27 #include <time.h>
28 #endif
29
30 #ifdef _USING_FCGI_
31 #include "boinc_fcgi.h"
32 #endif
33
34 #include "error_numbers.h"
35 #include "parse.h"
36 #include "util.h"
37
38 #include "prefs.h"
39
40 GLOBAL_PREFS_MASK::GLOBAL_PREFS_MASK() {
41     clear();
42 }
43
44 void GLOBAL_PREFS_MASK::clear() {
45     memset(this, 0, sizeof(GLOBAL_PREFS_MASK));
46 }
47
48 void GLOBAL_PREFS_MASK::set_all() {
49     battery_charge_min_pct = true;
50     battery_max_temperature = true;
51     confirm_before_connecting = true;
52     cpu_scheduling_period_minutes = true;
53     cpu_usage_limit = true;
54     daily_xfer_limit_mb = true;
55     daily_xfer_period_days = true;
56     disk_interval = true;
57     disk_max_used_gb = true;
58     disk_max_used_pct = true;
59     disk_min_free_gb = true;
60     dont_verify_images = true;
61     end_hour = true;
62     hangup_if_dialed = true;
63     idle_time_to_run = true;
64     leave_apps_in_memory = true;
65     max_bytes_sec_down = true;
66     max_bytes_sec_up = true;
```

```
67      max_ncpus= true;
68      max_ncpus_pct = true;
69      net_end_hour = true;
70      net_start_hour = true;
71      network_wifi_only = true;
72      ram_max_used_busy_frac = true;
73      ram_max_used_idle_frac = true;
74      run_gpu_if_user_active = true;
75      run_if_user_active = true;
76      run_on_batteries = true;
77      start_hour = true;
78      suspend_cpu_usage = 0;
79      suspend_if_no_recent_input = true;
80      vm_max_used_frac = true;
81      work_buf_additional_days = true;
82      work_buf_min_days = true;
83    store_projects_external_disk = true;
84  }
85
86  bool GLOBAL_PREFS_MASK::are_prefs_set() {
87      if (battery_charge_min_pct) return true;
88      if (battery_max_temperature) return true;
89      if (confirm_before_connecting) return true;
90      if (cpu_scheduling_period_minutes) return true;
91      if (cpu_usage_limit) return true;
92      if (daily_xfer_limit_mb) return true;
93      if (daily_xfer_period_days) return true;
94      if (disk_interval) return true;
95      if (disk_max_used_gb) return true;
96      if (disk_max_used_pct) return true;
97      if (disk_min_free_gb) return true;
98      if (dont_verify_images) return true;
99      if (end_hour) return true;
100     if (hangup_if_dialed) return true;
101     if (idle_time_to_run) return true;
102     if (leave_apps_in_memory) return true;
103     if (max_bytes_sec_down) return true;
104     if (max_bytes_sec_up) return true;
105     if (max_ncpus) return true;
106     if (max_ncpus_pct) return true;
107     if (net_start_hour) return true;
108     if (network_wifi_only) return true;
109     if (net_end_hour) return true;
110     if (ram_max_used_busy_frac) return true;
111     if (ram_max_used_idle_frac) return true;
112     if (run_gpu_if_user_active) return true;
113     if (run_if_user_active) return true;
114     if (run_on_batteries) return true;
115     if (start_hour) return true;
116     if (suspend_if_no_recent_input) return true;
117     if (suspend_cpu_usage) return true;
118     if (vm_max_used_frac) return true;
119     if (work_buf_additional_days) return true;
120     if (work_buf_min_days) return true;
121     return false;
122 }
123
124 bool GLOBAL_PREFS_MASK::are_simple_prefs_set() {
```

```
125     if (cpu_usage_limit) return true;
126     if (disk_max_used_gb) return true;
127     if (end_hour) return true;
128     if (idle_time_to_run) return true;
129     if (net_end_hour) return true;
130     if (net_start_hour) return true;
131     if (run_if_user_active) return true;
132     if (run_on_batteries) return true;
133     if (start_hour) return true;
134     return false;
135 }
136 
137 
138 // TIME_SPAN implementation
139 
140 bool TIME_SPAN::suspended(double hour) const {
141     if (start_hour == end_hour) return false;
142     if (start_hour == 0 && end_hour == 24) return false;
143     if (start_hour == 24 && end_hour == 0) return true;
144     if (start_hour < end_hour) {
145         return (hour < start_hour || hour > end_hour);
146     } else {
147         return (hour >= end_hour && hour < start_hour);
148     }
149 }
150 
151 TIME_SPAN::TimeMode TIME_SPAN::mode() const {
152     if (end_hour == start_hour || (start_hour == 0 && end_hour == 24)) {
153         return Always;
154     } else if (start_hour == 24 && end_hour == 0) {
155         return Never;
156     }
157     return Between;
158 }
159 
160 
161 // TIME_PREFS implementation
162 
163 void TIME_PREFS::clear() {
164     start_hour = 0;
165     end_hour = 0;
166     week.clear();
167 }
168 
169 bool TIME_PREFS::suspended(double now) {
170     time_t t = (time_t)now;
171     struct tm* tmp = localtime(&t);
172     double hour = (tmp->tm_hour * 3600 + tmp->tm_min * 60 + tmp->tm_sec)
        / 3600.;
173     int day = tmp->tm_wday;
174 
175     // Use day-specific settings, if they exist:
176     //
177     if (day>=0 && day<7 && week.days[day].present) {
178         return week.days[day].suspended(hour);
179     }
180     return TIME_SPAN::suspended(hour);
181 }
```

```
182
183
184 // WEEK_PREFS implementation
185
186
187 void WEEK_PREFS::set(int day, double start, double end) {
188     if (day < 0 || day > 6) return;
189     days[day].present = true;
190     days[day].start_hour = start;
191     days[day].end_hour = end;
192 }
193
194
195 void WEEK_PREFS::set(int day, TIME_SPAN* time) {
196     if (day < 0 || day > 6) return;
197     days[day].present = true;
198     days[day].start_hour = time->start_hour;
199     days[day].end_hour = time->end_hour;
200 }
201
202 void WEEK_PREFS::unset(int day) {
203     if (day < 0 || day > 6) return;
204     days[day].present = false;
205 }
206
207 // The following values determine how the client behaves
208 // if there are no global prefs (e.g. on our very first RPC).
209 // These should impose minimal restrictions,
210 // so that the client can do the RPC and get the global prefs from the
    server
211 //
212 void GLOBAL_PREFS::defaults() {
213     battery_charge_min_pct = 95;
214     battery_max_temperature = 45;
215     confirm_before_connecting = true;
216     cpu_scheduling_period_minutes = 60;
217     cpu_times.clear();
218     cpu_usage_limit = 100;
219     daily_xfer_limit_mb = 0;
220     daily_xfer_period_days = 0;
221     disk_interval = 60;
222     disk_max_used_gb = 1000;
223     disk_max_used_pct = 90;
224     disk_min_free_gb = 0.1;
225     dont_verify_images = false;
226     hangup_if_dialed = false;
227     idle_time_to_run = 3;
228     leave_apps_in_memory = false;
229     max_bytes_sec_down = 0;
230     max_bytes_sec_up = 0;
231     max_ncpus = 0;
232     max_ncpus_pct = 0;
233     net_times.clear();
234     network_wifi_only = false;
235     ram_max_used_busy_frac = 0.5;
236     ram_max_used_idle_frac = 0.9;
237     run_gpu_if_user_active = false;
238     run_if_user_active = true;
```

```
239     run_on_batteries = true;
240     suspend_cpu_usage = 25;
241     suspend_if_no_recent_input = 0;
242     vm_max_used_frac = 0.75;
243     work_buf_additional_days = 0.5;
244     work_buf_min_days = 0.1;
245   store_projects_external_disk = false;
246     // don't initialize source_project, source_scheduler,
247     // mod_time, host_specific here
248     // since they are outside of <venue> elements,
249     // and this is called when find the right venue.
250     // Also, don't memset to 0
251 }
252
253 // before parsing
254 void GLOBAL_PREFS::clear_bools() {
255     run_on_batteries = false;
256     run_if_user_active = false;
257     leave_apps_in_memory = false;
258     confirm_before_connecting = false;
259     hangup_if_dialed = false;
260     dont_verify_images = false;
261     network_wifi_only = true;
262   store_projects_external_disk = false;
263 }
264
265 void GLOBAL_PREFS::init() {
266     defaults();
267     strcpy(source_project, "");
268     strcpy(source_scheduler, "");
269     mod_time = 0;
270     host_specific = false;
271 }
272
273 GLOBAL_PREFS::GLOBAL_PREFS() {
274     init();
275 }
276
277 // Parse XML global prefs, setting defaults first.
278 //
279 int GLOBAL_PREFS::parse(
280     XML_PARSER& xp, const char* host_venue, bool& found_venue,
        GLOBAL_PREFS_MASK& mask
281 ) {
282     init();
283     clear_bools();
284     return parse_override(xp, host_venue, found_venue, mask);
285 }
286
287 int GLOBAL_PREFS::parse_day(XML_PARSER& xp) {
288     int day_of_week = -1;
289     bool has_cpu = false;
290     bool has_net = false;
291     double start_hour = 0;
292     double end_hour = 0;
293     double net_start_hour = 0;
294     double net_end_hour = 0;
295
```

```
296     while (!xp.get_tag()) {
297         if (!xp.is_tag) continue;
298         if (xp.match_tag("/day_prefs")) {
299             if (day_of_week < 0 || day_of_week > 6) return ERR_XML_PARSE;
300             if (has_cpu) {
301                 cpu_times.week.set(day_of_week, start_hour, end_hour);
302             }
303             if (has_net) {
304                 net_times.week.set(day_of_week, net_start_hour,
                        net_end_hour);
305             }
306             return 0;
307         }
308         if (xp.parse_int("day_of_week", day_of_week)) continue;
309         if (xp.parse_double("start_hour", start_hour)) {
310             has_cpu = true;
311             continue;
312         }
313         if (xp.parse_double("end_hour", end_hour)) {
314             has_cpu = true;
315             continue;
316         }
317         if (xp.parse_double("net_start_hour", net_start_hour)) {
318             has_net = true;
319             continue;
320         }
321         if (xp.parse_double("net_end_hour", net_end_hour)) {
322             has_net = true;
323             continue;
324         }
325         xp.skip_unexpected(true, "GLOBAL_PREFS::parse_day");
326     }
327     return ERR_XML_PARSE;
328 }
329
330
331 // Parse global prefs, overriding whatever is currently in the structure.
332 //
333 // If host_venue is nonempty and we find an element of the form
334 // <venue name="X">
335 //   ...
336 // </venue>
337 // where X==host_venue, then parse that and ignore the rest.
338 // Otherwise ignore <venue> elements.
339 //
340 // The start tag may or may not have already been parsed
341 //
342 int GLOBAL_PREFS::parse_override(
343     XML_PARSER& xp, const char* host_venue, bool& found_venue,
        GLOBAL_PREFS_MASK& mask
344 ) {
345     char buf2[256], attrs[256];
346     bool in_venue = false, in_correct_venue=false;
347     double dtemp;
348     int itemp;
349
350     found_venue = false;
351     mask.clear();
```

```
352
353     while (!xp.get_tag(attrs, sizeof(attrs))) {
354         if (!xp.is_tag) continue;
355         if (xp.match_tag("global_preferences")) continue;
356         if (xp.match_tag("/global_preferences")) {
357             return 0;
358         }
359         if (in_venue) {
360             if (xp.match_tag("/venue")) {
361                 if (in_correct_venue) {
362                     return 0;
363                 } else {
364                     in_venue = false;
365                     continue;
366                 }
367             } else {
368                 if (!in_correct_venue) continue;
369             }
370         } else {
371             if (strstr(xp.parsed_tag, "venue")) {
372                 in_venue = true;
373                 parse_attr(attrs, "name", buf2, sizeof(buf2));
374                 if (!strcmp(buf2, host_venue)) {
375                     defaults();
376                     clear_bools();
377                     mask.clear();
378                     in_correct_venue = true;
379                     found_venue = true;
380                 } else {
381                     in_correct_venue = false;
382                 }
383                 continue;
384             }
385         }
386         if (xp.parse_str("source_project", source_project, sizeof(
                source_project))) continue;
387         if (xp.parse_str("source_scheduler", source_scheduler, sizeof(
                source_scheduler))) {
388             continue;
389         }
390         if (xp.parse_double("mod_time", mod_time)) {
391             double now = dtime();
392             if (mod_time > now) {
393                 mod_time = now;
394             }
395             continue;
396         }
397         if (xp.parse_double("battery_charge_min_pct",
                battery_charge_min_pct)) {
398             mask.battery_charge_min_pct = true;
399             continue;
400         }
401         if (xp.parse_double("battery_max_temperature",
                battery_max_temperature)) {
402             mask.battery_max_temperature = true;
403             continue;
404         }
405         if (xp.parse_bool("run_on_batteries", run_on_batteries)) {
```

```
406             mask.run_on_batteries = true;
407             continue;
408         }
409         if (xp.parse_bool("run_if_user_active", run_if_user_active)) {
410             mask.run_if_user_active = true;
411             continue;
412         }
413         if (xp.parse_bool("run_gpu_if_user_active",
                run_gpu_if_user_active)) {
414             mask.run_gpu_if_user_active = true;
415             continue;
416         }
417         if (xp.parse_double("idle_time_to_run", idle_time_to_run)) {
418             mask.idle_time_to_run = true;
419             continue;
420         }
421         if (xp.parse_double("suspend_if_no_recent_input",
                suspend_if_no_recent_input)) {
422             mask.suspend_if_no_recent_input = true;
423             continue;
424         }
425         if (xp.parse_double("suspend_cpu_usage", suspend_cpu_usage)) {
426             mask.suspend_cpu_usage = true;
427             continue;
428         }
429         if (xp.parse_double("start_hour", cpu_times.start_hour)) {
430             mask.start_hour = true;
431             continue;
432         }
433         if (xp.parse_double("end_hour", cpu_times.end_hour)) {
434             mask.end_hour = true;
435             continue;
436         }
437         if (xp.parse_double("net_start_hour", net_times.start_hour)) {
438             mask.net_start_hour = true;
439             continue;
440         }
441         if (xp.parse_double("net_end_hour", net_times.end_hour)) {
442             mask.net_end_hour = true;
443             continue;
444         }
445         if (xp.match_tag("day_prefs")) {
446             parse_day(xp);
447             continue;
448         }
449         if (xp.parse_bool("leave_apps_in_memory", leave_apps_in_memory))
                {
450             mask.leave_apps_in_memory = true;
451             continue;
452         }
453         if (xp.parse_bool("confirm_before_connecting",
                confirm_before_connecting)) {
454             mask.confirm_before_connecting = true;
455             continue;
456         }
457         if (xp.parse_bool("hangup_if_dialed", hangup_if_dialed)) {
458             mask.hangup_if_dialed = true;
459             continue;
```

```
460             }
461             if (xp.parse_bool("dont_verify_images", dont_verify_images)) {
462                 mask.dont_verify_images = true;
463                 continue;
464             }
465             if (xp.parse_double("work_buf_min_days", work_buf_min_days)) {
466                 if (work_buf_min_days < 0) work_buf_min_days = 0;
467                 mask.work_buf_min_days = true;
468                 continue;
469             }
470             if (xp.parse_double("work_buf_additional_days",
                work_buf_additional_days)) {
471                 if (work_buf_additional_days < 0) work_buf_additional_days =
                    0;
472                 mask.work_buf_additional_days = true;
473                 continue;
474             }
475             if (xp.parse_double("max_ncpus_pct", max_ncpus_pct)) {
476                 if (max_ncpus_pct < 0) max_ncpus_pct = 0;
477                 if (max_ncpus_pct > 100) max_ncpus_pct = 100;
478                 mask.max_ncpus_pct = true;
479                 continue;
480             }
481             if (xp.parse_int("max_cpus", max_ncpus)) {
482                 if (max_ncpus < 0) max_ncpus = 0;
483                 mask.max_ncpus = true;
484                 continue;
485             }
486             if (xp.parse_double("disk_interval", disk_interval)) {
487                 if (disk_interval<0) disk_interval = 0;
488                 mask.disk_interval = true;
489                 continue;
490             }
491             if (xp.parse_double("cpu_scheduling_period_minutes",
                cpu_scheduling_period_minutes)) {
492                 if (cpu_scheduling_period_minutes < 0.0001)
                    cpu_scheduling_period_minutes = 60;
493                 mask.cpu_scheduling_period_minutes = true;
494                 continue;
495             }
496             if (xp.parse_double("disk_max_used_gb", disk_max_used_gb)) {
497                 mask.disk_max_used_gb = true;
498                 continue;
499             }
500             if (xp.parse_double("disk_max_used_pct", disk_max_used_pct)) {
501                 mask.disk_max_used_pct = true;
502                 continue;
503             }
504             if (xp.parse_double("disk_min_free_gb", disk_min_free_gb)) {
505                 mask.disk_min_free_gb = true;
506                 continue;
507             }
508             if (xp.parse_double("vm_max_used_pct", dtemp)) {
509                 vm_max_used_frac = dtemp/100;
510                 mask.vm_max_used_frac = true;
511                 continue;
512             }
513             if (xp.parse_double("ram_max_used_busy_pct", dtemp)) {
```

```
514             if (!dtemp) dtemp = 100;
515             ram_max_used_busy_frac = dtemp/100;
516             mask.ram_max_used_busy_frac = true;
517             continue;
518         }
519         if (xp.parse_double("ram_max_used_idle_pct", dtemp)) {
520             if (!dtemp) dtemp = 100;
521             ram_max_used_idle_frac = dtemp/100;
522             mask.ram_max_used_idle_frac = true;
523             continue;
524         }
525         if (xp.parse_double("max_bytes_sec_up", max_bytes_sec_up)) {
526             if (max_bytes_sec_up < 0) max_bytes_sec_up = 0;
527             mask.max_bytes_sec_up = true;
528             continue;
529         }
530         if (xp.parse_double("max_bytes_sec_down", max_bytes_sec_down)) {
531             if (max_bytes_sec_down < 0) max_bytes_sec_down = 0;
532             mask.max_bytes_sec_down = true;
533             continue;
534         }
535         if (xp.parse_double("cpu_usage_limit", dtemp)) {
536             if (dtemp > 0 && dtemp <= 100) {
537                 cpu_usage_limit = dtemp;
538                 mask.cpu_usage_limit = true;
539             }
540             continue;
541         }
542         if (xp.parse_double("daily_xfer_limit_mb", dtemp)) {
543             if (dtemp >= 0) {
544                 daily_xfer_limit_mb = dtemp;
545                 mask.daily_xfer_limit_mb = true;
546             }
547             continue;
548         }
549         if (xp.parse_int("daily_xfer_period_days", itemp)) {
550             if (itemp >= 0) {
551                 daily_xfer_period_days = itemp;
552                 mask.daily_xfer_period_days = true;
553             }
554             continue;
555         }
556         if (xp.parse_bool("network_wifi_only", network_wifi_only)) {
557             continue;
558         }
559         if (xp.parse_bool("host_specific", host_specific)) {
560             continue;
561         }
562     if (xp.parse_bool("store_projects_external_disk",
        store_projects_external_disk)) {
563       continue;
564     }
565         // false means don't print anything
566         xp.skip_unexpected(false, "GLOBAL_PREFS::parse_override");
567     }
568     return ERR_XML_PARSE;
569 }
570
```

```
571 // Parse global prefs file
572 //
573 int GLOBAL_PREFS::parse_file(
574     const char* filename, const char* host_venue, bool& found_venue
575 ) {
576     FILE* f;
577     GLOBAL_PREFS_MASK mask;
578     int retval;
579
580 #ifndef _USING_FCGI_
581     f = fopen(filename, "r");
582 #else
583     f = FCGI::fopen(filename, "r");
584 #endif
585     if (!f) return ERR_FOPEN;
586     MIOFILE mf;
587     mf.init_file(f);
588     XML_PARSER xp(&mf);
589     retval = parse(xp, host_venue, found_venue, mask);
590     fclose(f);
591     return retval;
592 }
593
594 // Write the global prefs that are actually in force
595 // (our particular venue, modified by overwrite file).
596 // This is used to write
597 // 1) the app init data file
598 // 2) GUI RPC get_state reply
599 // 3) scheduler request (<working_global_preferences> element)
600 //
601 int GLOBAL_PREFS::write(MIOFILE& f) {
602     f.printf(
603         "<global_preferences>\n"
604         "   <source_project>%s</source_project>\n"
605         "   <mod_time>%f</mod_time>\n"
606         "   <battery_charge_min_pct>%f</battery_charge_min_pct>\n"
607         "   <battery_max_temperature>%f</battery_max_temperature>\n"
608         "   <run_on_batteries>%d</run_on_batteries>\n"
609         "   <run_if_user_active>%d</run_if_user_active>\n"
610         "   <run_gpu_if_user_active>%d</run_gpu_if_user_active>\n"
611         "   <suspend_if_no_recent_input>%f</suspend_if_no_recent_input>\n
        "
612         "   <suspend_cpu_usage>%f</suspend_cpu_usage>\n"
613         "   <start_hour>%f</start_hour>\n"
614         "   <end_hour>%f</end_hour>\n"
615         "   <net_start_hour>%f</net_start_hour>\n"
616         "   <net_end_hour>%f</net_end_hour>\n"
617         "   <leave_apps_in_memory>%d</leave_apps_in_memory>\n"
618         "   <confirm_before_connecting>%d</confirm_before_connecting>\n"
619         "   <hangup_if_dialed>%d</hangup_if_dialed>\n"
620         "   <dont_verify_images>%d</dont_verify_images>\n"
621         "   <work_buf_min_days>%f</work_buf_min_days>\n"
622         "   <work_buf_additional_days>%f</work_buf_additional_days>\n"
623         "   <max_ncpus_pct>%f</max_ncpus_pct>\n"
624         "   <cpu_scheduling_period_minutes>%f</
        cpu_scheduling_period_minutes>\n"
625         "   <disk_interval>%f</disk_interval>\n"
626         "   <disk_max_used_gb>%f</disk_max_used_gb>\n"
```

```
627         "   <disk_max_used_pct>%f</disk_max_used_pct>\n"
628         "   <disk_min_free_gb>%f</disk_min_free_gb>\n"
629         "   <vm_max_used_pct>%f</vm_max_used_pct>\n"
630         "   <ram_max_used_busy_pct>%f</ram_max_used_busy_pct>\n"
631         "   <ram_max_used_idle_pct>%f</ram_max_used_idle_pct>\n"
632         "   <idle_time_to_run>%f</idle_time_to_run>\n"
633         "   <max_bytes_sec_up>%f</max_bytes_sec_up>\n"
634         "   <max_bytes_sec_down>%f</max_bytes_sec_down>\n"
635         "   <cpu_usage_limit>%f</cpu_usage_limit>\n"
636         "   <daily_xfer_limit_mb>%f</daily_xfer_limit_mb>\n"
637         "   <daily_xfer_period_days>%d</daily_xfer_period_days>\n"
638         "   <override_file_present>%d</override_file_present>\n"
639         "   <network_wifi_only>%d</network_wifi_only>\n"
640     "   <store_projects_external_disk>%d</store_projects_external_disk>\n
          ",
641         source_project,
642         mod_time,
643         battery_charge_min_pct,
644         battery_max_temperature,
645         run_on_batteries?1:0,
646         run_if_user_active?1:0,
647         run_gpu_if_user_active?1:0,
648         suspend_if_no_recent_input,
649         suspend_cpu_usage,
650         cpu_times.start_hour,
651         cpu_times.end_hour,
652         net_times.start_hour,
653         net_times.end_hour,
654         leave_apps_in_memory?1:0,
655         confirm_before_connecting?1:0,
656         hangup_if_dialed?1:0,
657         dont_verify_images?1:0,
658         work_buf_min_days,
659         work_buf_additional_days,
660         max_ncpus_pct,
661         cpu_scheduling_period_minutes,
662         disk_interval,
663         disk_max_used_gb,
664         disk_max_used_pct,
665         disk_min_free_gb,
666         vm_max_used_frac*100,
667         ram_max_used_busy_frac*100,
668         ram_max_used_idle_frac*100,
669         idle_time_to_run,
670         max_bytes_sec_up,
671         max_bytes_sec_down,
672         cpu_usage_limit,
673         daily_xfer_limit_mb,
674         daily_xfer_period_days,
675         override_file_present?1:0,
676         network_wifi_only?1:0
677     ,store_projects_external_disk?1:0
678     );
679     if (max_ncpus) {
680         f.printf("   <max_cpus>%d</max_cpus>\n", max_ncpus);
681     }
682 
683     write_day_prefs(f);
```

```
684
685     f.printf("</global_preferences>\n");
686
687     return 0;
688 }
689
690 void GLOBAL_PREFS::write_day_prefs(MIOFILE& f) {
691     for (int i=0; i<7; i++) {
692         bool cpu_present = cpu_times.week.days[i].present;
693         bool net_present = net_times.week.days[i].present;
694         //write only when needed
695         if (net_present || cpu_present) {
696
697             f.printf("    <day_prefs>\n");
698             f.printf("       <day_of_week>%d</day_of_week>\n", i);
699             if (cpu_present) {
700                 f.printf(
701                     "       <start_hour>%.02f</start_hour>\n"
702                     "       <end_hour>%.02f</end_hour>\n",
703                     cpu_times.week.days[i].start_hour,
704                     cpu_times.week.days[i].end_hour
705                 );
706             }
707             if (net_present) {
708                 f.printf(
709                     "       <net_start_hour>%.02f</net_start_hour>\n"
710                     "       <net_end_hour>%.02f</net_end_hour>\n",
711                     net_times.week.days[i].start_hour,
712                     net_times.week.days[i].end_hour
713                 );
714             }
715             f.printf("    </day_prefs>\n");
716         }
717     }
718 }
719
720 // write a subset of the global preferences,
721 // as selected by the mask of bools
722 //
723 int GLOBAL_PREFS::write_subset(MIOFILE& f, GLOBAL_PREFS_MASK& mask) {
724     if (!mask.are_prefs_set()) return 0;
725
726     f.printf("<global_preferences>\n");
727     if (mask.run_on_batteries) {
728         f.printf("    <run_on_batteries>%d</run_on_batteries>\n",
729             run_on_batteries?1:0
730         );
731     }
732     if (mask.run_if_user_active) {
733         f.printf("    <run_if_user_active>%d</run_if_user_active>\n",
734             run_if_user_active?1:0
735         );
736     }
737     if (mask.run_gpu_if_user_active) {
738         f.printf("    <run_gpu_if_user_active>%d</run_gpu_if_user_active>\
            n",
739             run_gpu_if_user_active?1:0
740         );
```

```
741     }
742     if (mask.idle_time_to_run) {
743         f.printf("    <idle_time_to_run>%f</idle_time_to_run>\n",
                idle_time_to_run);
744     }
745     if (mask.suspend_if_no_recent_input) {
746
747         f.printf("    <suspend_if_no_recent_input>%f</
                suspend_if_no_recent_input>\n",
748             suspend_if_no_recent_input
749         );
750     }
751     if (mask.suspend_cpu_usage) {
752
753         f.printf("    <suspend_cpu_usage>%f</suspend_cpu_usage>\n",
754             suspend_cpu_usage
755         );
756     }
757     if (mask.start_hour) {
758         f.printf("    <start_hour>%f</start_hour>\n", cpu_times.start_hour
                );
759     }
760     if (mask.end_hour) {
761         f.printf("    <end_hour>%f</end_hour>\n", cpu_times.end_hour);
762     }
763     if (mask.net_start_hour) {
764         f.printf("    <net_start_hour>%f</net_start_hour>\n", net_times.
                start_hour);
765     }
766     if (mask.net_end_hour) {
767         f.printf("    <net_end_hour>%f</net_end_hour>\n", net_times.
                end_hour);
768     }
769     if (mask.leave_apps_in_memory) {
770         f.printf("    <leave_apps_in_memory>%d</leave_apps_in_memory>\n",
771             leave_apps_in_memory?1:0
772         );
773     }
774     if (mask.battery_charge_min_pct) {
775         f.printf("    <battery_charge_min_pct>%f</battery_charge_min_pct>\
                n",
776             battery_charge_min_pct
777         );
778     }
779     if (mask.battery_max_temperature) {
780         f.printf("    <battery_max_temperature>%f</battery_max_temperature
                >\n",
781             battery_max_temperature
782         );
783     }
784     if (mask.confirm_before_connecting) {
785         f.printf("    <confirm_before_connecting>%d</
                confirm_before_connecting>\n",
786             confirm_before_connecting?1:0
787         );
788     }
789     if (mask.hangup_if_dialed) {
790         f.printf("    <hangup_if_dialed>%d</hangup_if_dialed>\n",
```

```
791             hangup_if_dialed?1:0
792         );
793     }
794     if (mask.dont_verify_images) {
795         f.printf("   <dont_verify_images>%d</dont_verify_images>\n",
796             dont_verify_images?1:0
797         );
798     }
799     if (mask.work_buf_min_days) {
800         f.printf("   <work_buf_min_days>%f</work_buf_min_days>\n",
            work_buf_min_days);
801     }
802     if (mask.work_buf_additional_days) {
803         f.printf("   <work_buf_additional_days>%f</
            work_buf_additional_days>\n", work_buf_additional_days);
804     }
805     if (mask.max_ncpus_pct) {
806         f.printf("   <max_ncpus_pct>%f</max_ncpus_pct>\n", max_ncpus_pct)
            ;
807     }
808     if (mask.max_ncpus) {
809         f.printf("   <max_cpus>%d</max_cpus>\n", max_ncpus);
810     }
811     if (mask.cpu_scheduling_period_minutes) {
812         f.printf("   <cpu_scheduling_period_minutes>%f</
            cpu_scheduling_period_minutes>\n",
            cpu_scheduling_period_minutes);
813     }
814     if (mask.disk_interval) {
815         f.printf("   <disk_interval>%f</disk_interval>\n", disk_interval)
            ;
816     }
817     if (mask.disk_max_used_gb) {
818         f.printf("   <disk_max_used_gb>%f</disk_max_used_gb>\n",
            disk_max_used_gb);
819     }
820     if (mask.disk_max_used_pct) {
821         f.printf("   <disk_max_used_pct>%f</disk_max_used_pct>\n",
            disk_max_used_pct);
822     }
823     if (mask.disk_min_free_gb) {
824         f.printf("   <disk_min_free_gb>%f</disk_min_free_gb>\n",
            disk_min_free_gb);
825     }
826     if (mask.vm_max_used_frac) {
827         f.printf("   <vm_max_used_pct>%f</vm_max_used_pct>\n",
            vm_max_used_frac*100);
828     }
829     if (mask.ram_max_used_busy_frac) {
830         f.printf("   <ram_max_used_busy_pct>%f</ram_max_used_busy_pct>\n"
            , ram_max_used_busy_frac*100);
831     }
832     if (mask.ram_max_used_idle_frac) {
833         f.printf("   <ram_max_used_idle_pct>%f</ram_max_used_idle_pct>\n"
            , ram_max_used_idle_frac*100);
834     }
835     if (mask.max_bytes_sec_up) {
836         f.printf("   <max_bytes_sec_up>%f</max_bytes_sec_up>\n",
```

```
            max_bytes_sec_up);
837     }
838     if (mask.max_bytes_sec_down) {
839         f.printf("   <max_bytes_sec_down>%f</max_bytes_sec_down>\n",
            max_bytes_sec_down);
840     }
841     if (mask.cpu_usage_limit) {
842         f.printf("   <cpu_usage_limit>%f</cpu_usage_limit>\n",
            cpu_usage_limit);
843     }
844     if (mask.daily_xfer_limit_mb) {
845         f.printf("   <daily_xfer_limit_mb>%f</daily_xfer_limit_mb>\n",
            daily_xfer_limit_mb);
846     }
847     if (mask.daily_xfer_period_days) {
848         f.printf("   <daily_xfer_period_days>%d</daily_xfer_period_days>\
            n", daily_xfer_period_days);
849     }
850     if (mask.network_wifi_only) {
851         f.printf("   <network_wifi_only>%d</network_wifi_only>\n",
            network_wifi_only?1:0 );
852   }
853
854   if (mask.store_projects_external_disk) {
855     f.printf("   <store_projects_external_disk>%d</
          store_projects_external_disk>\n", store_projects_external_disk ?
          1:0 );
856   }
857
858     write_day_prefs(f);
859     f.printf("</global_preferences>\n");
860     return 0;
861 }
```

A.5 PrefsActivity.java

Listagem de Código A.5: PrefsActivity.java

```
1  /*****************************************************************
2   * This file is part of BOINC.
3   * http://boinc.berkeley.edu
4   * Copyright (C) 2012 University of California
5   *
6   * BOINC is free software; you can redistribute it and/or modify it
7   * under the terms of the GNU Lesser General Public License
8   * as published by the Free Software Foundation,
9   * either version 3 of the License, or (at your option) any later version
        .
10  *
11  * BOINC is distributed in the hope that it will be useful,
12  * but WITHOUT ANY WARRANTY; without even the implied warranty of
13  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
14  * See the GNU Lesser General Public License for more details.
15  *
16  * You should have received a copy of the GNU Lesser General Public
        License
17  * along with BOINC.  If not, see <http://www.gnu.org/licenses/>.
```

```
18  ********************************************************************/
19 package edu.berkeley.boinc;
20
21 import edu.berkeley.boinc.utils.*;
22
23 import java.util.ArrayList;
24 import edu.berkeley.boinc.adapter.PrefsListAdapter;
25 import edu.berkeley.boinc.adapter.PrefsListItemWrapper;
26 import edu.berkeley.boinc.adapter.PrefsListItemWrapperBool;
27 import edu.berkeley.boinc.adapter.PrefsListItemWrapperValue;
28 import edu.berkeley.boinc.adapter.PrefsLogOptionsListAdapter;
29 import edu.berkeley.boinc.client.ClientNotification;
30 import edu.berkeley.boinc.client.ClientStatus;
31 import edu.berkeley.boinc.client.Monitor;
32 import edu.berkeley.boinc.rpc.GlobalPreferences;
33 import edu.berkeley.boinc.rpc.HostInfo;
34 import android.app.Dialog;
35 import android.content.ComponentName;
36 import android.content.Intent;
37 import android.content.ServiceConnection;
38 import android.os.AsyncTask;
39 import android.os.Bundle;
40 import android.os.IBinder;
41 import android.support.v4.app.FragmentActivity;
42 import android.util.Log;
43 import android.view.View;
44 import android.view.Window;
45 import android.view.View.OnClickListener;
46 import android.widget.Button;
47 import android.widget.CheckBox;
48 import android.widget.EditText;
49 import android.widget.ListView;
50 import android.widget.SeekBar;
51 import android.widget.TextView;
52 import android.widget.Toast;
53
54 public class PrefsActivity extends FragmentActivity {
55
56   private Monitor monitor;
57   private Boolean mIsBound = false;
58
59   private ListView lv;
60   private PrefsListAdapter listAdapter;
61
62   private ArrayList<PrefsListItemWrapper> data = new ArrayList<
       PrefsListItemWrapper>(); //Adapter for list data
63   private GlobalPreferences clientPrefs = null; //preferences of the
       client, read on every onResume via RPC
64   private AppPreferences appPrefs = null; //Android specific preferences,
        singleton of monitor
65   private HostInfo hostinfo = null;
66
67   public void onCreate(Bundle savedInstanceState) {
68     super.onCreate(savedInstanceState);
69     doBindService();
70   }
71
72   /*
```

```
73    * Service binding part
74    * only necessary, when function on monitor instance has to be called
75    */
76   private ServiceConnection mConnection = new ServiceConnection() {
77       public void onServiceConnected(ComponentName className, IBinder
             service) {
78         if(Logging.DEBUG) Log.d(Logging.TAG,"PrefsActivity
              onServiceConnected");
79           monitor = ((Monitor.LocalBinder)service).getService();
80         mIsBound = true;
81       appPrefs = Monitor.getAppPrefs();
82       if(Logging.DEBUG) Log.d(Logging.TAG, "appPrefs available");
83       populateLayout();
84       }
85
86       public void onServiceDisconnected(ComponentName className) {
87           monitor = null;
88           mIsBound = false;
89       }
90   };
91
92   private void doBindService() {
93     if(!mIsBound) {
94       getApplicationContext().bindService(new Intent(this, Monitor.class)
             , mConnection, 0); //calling within Tab needs
             getApplicationContext() for bindService to work!
95     }
96   }
97
98   private void doUnbindService() {
99       if (mIsBound) {
100        getApplicationContext().unbindService(mConnection);
101          mIsBound = false;
102      }
103  }
104
105  private Boolean getPrefs() {
106    // try to get current client status from monitor
107    ClientStatus status;
108    try{
109      status  = Monitor.getClientStatus();
110    } catch (Exception e){
111      if(Logging.WARNING) Log.w(Logging.TAG,"PrefsActivity: Could not
           load data, clientStatus not initialized.");
112      return false;
113    }
114    clientPrefs = status.getPrefs(); //read prefs from client via rpc
115    if(clientPrefs == null) {
116      if(Logging.DEBUG) Log.d(Logging.TAG, "readPrefs: null, return false
           ");
117      return false;
118    }
119    //if(Logging.DEBUG) Log.d(Logging.TAG, "readPrefs done");
120    return true;
121  }
122
123  private Boolean getHostInfo() {
124    // try to get current client status from monitor
```

```
125     ClientStatus status;
126     try{
127       status  = Monitor.getClientStatus();
128     } catch (Exception e){
129       if(Logging.WARNING) Log.w(Logging.TAG,"PrefsActivity: Could not
            load data, clientStatus not initialized.");
130       return false;
131     }
132     hostinfo = status.getHostInfo(); //Get the hostinfo from client via
          rpc
133     if(hostinfo == null) {
134       if(Logging.DEBUG) Log.d(Logging.TAG, "getHostInfo: null, return
            false");
135       return false;
136     }
137     return true;
138   }
139
140   private void populateLayout() {
141
142     if(!getPrefs() || appPrefs == null || !getHostInfo()) {
143       if(Logging.DEBUG) Log.d(Logging.TAG, "populateLayout returns, data
            is not present");
144       setLayoutLoading();
145       return;
146     }
147
148     // setup layout
149     setContentView(R.layout.prefs_layout);
150     lv = (ListView) findViewById(R.id.listview);
151         listAdapter = new PrefsListAdapter(PrefsActivity.this,R.id.
              listview,data);
152         lv.setAdapter(listAdapter);
153
154     data.clear();
155
156     Boolean advanced = appPrefs.getShowAdvanced();
157
158     // general
159       data.add(new PrefsListItemWrapper(this,R.string.
            prefs_category_general,true));
160     data.add(new PrefsListItemWrapperBool(this,R.string.
          prefs_autostart_header,R.string.prefs_category_general,appPrefs.
          getAutostart()));
161     data.add(new PrefsListItemWrapperBool(this,R.string.
          prefs_show_notification_header,R.string.prefs_category_general,
          appPrefs.getShowNotification()));
162     data.add(new PrefsListItemWrapperBool(this,R.string.
          prefs_show_advanced_header,R.string.prefs_category_general,
          appPrefs.getShowAdvanced()));
163     // network
164       data.add(new PrefsListItemWrapper(this,R.string.
            prefs_category_network,true));
165     data.add(new PrefsListItemWrapperBool(this,R.string.
          prefs_network_wifi_only_header,R.string.prefs_category_network,
          clientPrefs.network_wifi_only));
166     if(advanced) data.add(new PrefsListItemWrapperValue(this,R.string.
          prefs_network_daily_xfer_limit_mb_header,R.string.
```

```
            prefs_category_network,clientPrefs.daily_xfer_limit_mb));
167         // power
168       data.add(new PrefsListItemWrapper(this,R.string.prefs_category_power,
              true));
169       data.add(new PrefsListItemWrapperBool(this,R.string.
              prefs_run_on_battery_header,R.string.prefs_category_power,
              clientPrefs.run_on_batteries));
170       data.add(new PrefsListItemWrapperValue(this,R.string.
              battery_charge_min_pct_header,R.string.prefs_category_power,
              clientPrefs.battery_charge_min_pct));
171       if(advanced) data.add(new PrefsListItemWrapperValue(this,R.string.
              battery_temperature_max_header,R.string.prefs_category_power,
              clientPrefs.battery_max_temperature));
172       // cpu
173       if(advanced) data.add(new PrefsListItemWrapper(this,R.string.
              prefs_category_cpu,true));
174       if(advanced && hostinfo.p_ncpus > 1) data.add(new
              PrefsListItemWrapperValue(this,R.string.
              prefs_cpu_number_cpus_header,R.string.prefs_category_cpu,
              pctCpuCoresToNumber(clientPrefs.max_ncpus_pct)));
175       if(advanced) data.add(new PrefsListItemWrapperValue(this,R.string.
              prefs_cpu_time_max_header,R.string.prefs_category_cpu,clientPrefs
              .cpu_usage_limit));
176       if(advanced) data.add(new PrefsListItemWrapperValue(this,R.string.
              prefs_cpu_other_load_suspension_header,R.string.
              prefs_category_cpu,clientPrefs.suspend_cpu_usage));
177       // storage
178       if (ExternalStorageUtil.isExternalStorageWritable() || advanced)
179         data.add(new PrefsListItemWrapper(this,R.string.
                prefs_category_storage,true));
180       if (ExternalStorageUtil.isExternalStorageWritable())
181         data.add(new PrefsListItemWrapperBool(this, R.string.
                prefs_store_projects_external_disk, R.string.
                prefs_category_storage,clientPrefs.store_projects_external_disk
                ));
182       if(advanced) data.add(new PrefsListItemWrapperValue(this,R.string.
              prefs_disk_max_pct_header,R.string.prefs_category_storage,
              clientPrefs.disk_max_used_pct));
183       if(advanced) data.add(new PrefsListItemWrapperValue(this,R.string.
              prefs_disk_min_free_gb_header,R.string.prefs_category_storage,
              clientPrefs.disk_min_free_gb));
184       // memory
185       if(advanced) data.add(new PrefsListItemWrapper(this,R.string.
              prefs_category_memory,true));
186       if(advanced) data.add(new PrefsListItemWrapperValue(this,R.string.
              prefs_memory_max_idle_header,R.string.prefs_category_memory,
              clientPrefs.ram_max_used_idle_frac));
187       // debug
188       if(advanced) data.add(new PrefsListItemWrapper(this,R.string.
              prefs_category_debug,true));
189       if(advanced) data.add(new PrefsListItemWrapper(this,R.string.
              prefs_client_log_flags_header,R.string.prefs_category_debug));
190       if(advanced) data.add(new PrefsListItemWrapperValue(this,R.string.
              prefs_gui_log_level_header,R.string.prefs_category_debug,(double)
              appPrefs.getLogLevel()));
191     }
192
193     private void updateLayout(){
```

```
194     listAdapter.notifyDataSetChanged();
195   }
196
197   // updates list item of boolean preference
198   // requires updateLayout to be called afterwards
199   private void updateBoolPref(int ID, Boolean newValue) {
200     if(Logging.DEBUG) Log.d(Logging.TAG, "updateBoolPref for ID: " + ID +
          " value: " + newValue);
201     for (PrefsListItemWrapper item: data) {
202       if(item.ID == ID){
203         ((PrefsListItemWrapperBool) item).setStatus(newValue);
204         continue;
205       }
206     }
207   }
208
209   // updates list item of value preference
210   // requires updateLayout to be called afterwards
211   private void updateValuePref(int ID, Double newValue) {
212     if(Logging.DEBUG) Log.d(Logging.TAG, "updateValuePref for ID: " + ID
         + " value: " + newValue);
213     for (PrefsListItemWrapper item: data) {
214       if(item.ID == ID){
215         ((PrefsListItemWrapperValue) item).status = newValue;
216         continue;
217       }
218     }
219   }
220
221   private void setLayoutLoading() {
222     if(Logging.DEBUG) Log.d(Logging.TAG,"setLayoutLoading()");
223         setContentView(R.layout.generic_layout_loading);
224         TextView loadingHeader = (TextView)findViewById(R.id.
              loading_header);
225         loadingHeader.setText(R.string.prefs_loading);
226   }
227
228   // onClick of listview items with prefs_layout_listitem_bool
229   public void onCbClick (View view) {
230     if(Logging.DEBUG) Log.d(Logging.TAG,"onCbClick");
231     Integer ID = (Integer) view.getTag();
232     CheckBox source = (CheckBox) view;
233     Boolean isSet = source.isChecked();
234
235     switch (ID) {
236     case R.string.prefs_autostart_header: //app pref
237       appPrefs.setAutostart(isSet);
238       updateBoolPref(ID, isSet);
239       updateLayout();
240       break;
241     case R.string.prefs_show_notification_header: //app pref
242       appPrefs.setShowNotification(isSet);
243       ClientNotification.getInstance(getApplicationContext()).update();
            // update notification
244       updateBoolPref(ID, isSet);
245       updateLayout();
246       break;
247     case R.string.prefs_show_advanced_header: //app pref
```

```
248       appPrefs.setShowAdvanced(isSet);
249       // reload complete layout to remove/add advanced elements
250       populateLayout();
251       break;
252     case R.string.prefs_run_on_battery_header: //client pref
253       clientPrefs.run_on_batteries = isSet;
254       updateBoolPref(ID, isSet);
255       new WriteClientPrefsAsync().execute(clientPrefs); //async task
              triggers layout update
256       break;
257     case R.string.prefs_network_wifi_only_header: //client pref
258       clientPrefs.network_wifi_only = isSet;
259       updateBoolPref(ID, isSet);
260       new WriteClientPrefsAsync().execute(clientPrefs); //async task
              triggers layout update
261       break;
262         case R.string.prefs_store_projects_external_disk:
263             clientPrefs.store_projects_external_disk = isSet;
264             updateBoolPref(ID, isSet);
265             new WriteClientPrefsAsync().execute(clientPrefs); //async
                  task triggers layout update
266             break;
267     }
268   }
269
270   // onClick of listview items with prefs_layout_listitem
271   public void onItemClick (View view) {
272     final Dialog dialog = new Dialog(this);
273     dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
274     Object item = view.getTag();
275
276     if(item instanceof PrefsListItemWrapperValue) {
277       final PrefsListItemWrapperValue valueWrapper = (
              PrefsListItemWrapperValue) item;
278       if(Logging.DEBUG) Log.d(Logging.TAG,"PrefsActivity onItemClick
              Value " + valueWrapper.ID);
279
280       if(valueWrapper.isPct) {
281         // show dialog with slider
282         dialog.setContentView(R.layout.prefs_layout_dialog_pct);
283         // setup slider
284         TextView sliderProgress = (TextView) dialog.findViewById(R.id.
              seekbar_status);
285         sliderProgress.setText(valueWrapper.status.intValue() + " " +
              getString(R.string.prefs_unit_pct));
286         Double seekBarDefault = valueWrapper.status / 10;
287         SeekBar slider = (SeekBar) dialog.findViewById(R.id.seekbar);
288         slider.setProgress(seekBarDefault.intValue());
289         slider.setOnSeekBarChangeListener(new SeekBar.
              OnSeekBarChangeListener() {
290             public void onProgressChanged(SeekBar seekBar, int progress
                  , boolean fromUser){
291               String progressString = (progress * 10) + " " + getString
                    (R.string.prefs_unit_pct);
292               TextView sliderProgress = (TextView) dialog.findViewById(
                    R.id.seekbar_status);
293                 sliderProgress.setText(progressString);
294             }
```

```
295             @Override
296             public void onStartTrackingTouch(SeekBar seekBar) {}
297             @Override
298             public void onStopTrackingTouch(SeekBar seekBar) {}
299             });
300         } else if(valueWrapper.isNumber) {
301           if(!getHostInfo()) {
302             if(Logging.WARNING) Log.w(Logging.TAG, "onItemClick missing
                  hostInfo");
303             return;
304           }
305
306           // show dialog with slider
307           dialog.setContentView(R.layout.prefs_layout_dialog_pct);
308           TextView sliderProgress = (TextView) dialog.findViewById(R.id.
                seekbar_status);
309           sliderProgress.setText(""+valueWrapper.status.intValue());
310           SeekBar slider = (SeekBar) dialog.findViewById(R.id.seekbar);
311
312           // slider setup depending on actual preference
313           if(valueWrapper.ID == R.string.prefs_cpu_number_cpus_header) {
314             slider.setMax(hostinfo.p_ncpus);
315             slider.setProgress(valueWrapper.status.intValue());
316             slider.setOnSeekBarChangeListener(new SeekBar.
                    OnSeekBarChangeListener() {
317               public void onProgressChanged(SeekBar seekBar, int progress,
                      boolean fromUser){
318                 if(progress == 0) progress = 1; // do not allow 0 cpus
319                 String progressString = String.valueOf(progress);
320                 TextView sliderProgress = (TextView) dialog.findViewById(R.
                        id.seekbar_status);
321                 sliderProgress.setText(progressString);
322               }
323               @Override
324               public void onStartTrackingTouch(SeekBar seekBar) {}
325               @Override
326               public void onStopTrackingTouch(SeekBar seekBar) {}
327             });
328           } else if (valueWrapper.ID == R.string.prefs_gui_log_level_header
                  ){
329             slider.setMax(5);
330             slider.setProgress(valueWrapper.status.intValue());
331             slider.setOnSeekBarChangeListener(new SeekBar.
                    OnSeekBarChangeListener() {
332               public void onProgressChanged(SeekBar seekBar, int progress,
                      boolean fromUser){
333                 String progressString = String.valueOf(progress);
334                 TextView sliderProgress = (TextView) dialog.findViewById(R.
                        id.seekbar_status);
335                 sliderProgress.setText(progressString);
336               }
337               @Override
338               public void onStartTrackingTouch(SeekBar seekBar) {}
339               @Override
340               public void onStopTrackingTouch(SeekBar seekBar) {}
341             });
342           }
343         } else {
```

```
344         // show dialog with edit text
345         dialog.setContentView(R.layout.prefs_layout_dialog);
346       }
347       // show preference name
348       ((TextView)dialog.findViewById(R.id.pref)).setText(valueWrapper.ID)
              ;
349
350       // setup buttons
351       Button confirm = (Button) dialog.findViewById(R.id.confirm);
352       confirm.setOnClickListener(new OnClickListener() {
353         @Override
354         public void onClick(View v) {
355                   double value = 0.0;
356                   Boolean clientPref = true;
357                   if(valueWrapper.isPct) {
358                     SeekBar slider = (SeekBar) dialog.findViewById(R.id.
                          seekbar);
359                     value = slider.getProgress()*10;
360                   } else if(valueWrapper.isNumber) {
361                     SeekBar slider = (SeekBar) dialog.findViewById(R.id.
                          seekbar);
362                     int sbProgress = slider.getProgress();
363                     if(valueWrapper.ID == R.string.
                          prefs_cpu_number_cpus_header) {
364                       if(sbProgress == 0) sbProgress = 1;
365                       value = numberCpuCoresToPct(sbProgress);
366                     } else if (valueWrapper.ID == R.string.
                          prefs_gui_log_level_header){
367                       appPrefs.setLogLevel(sbProgress);
368                       updateValuePref(valueWrapper.ID, (double) sbProgress
                            );
369                       clientPref = false; // avoid writing client prefs
                            via rpc
370                       updateLayout();
371                     }
372                   } else {
373                     EditText edit = (EditText) dialog.findViewById(R.id.
                          Input);
374                     String input = edit.getText().toString();
375                     Double valueTmp = parseInputValueToDouble(input);
376                     if(valueTmp == null) return;
377                     value = valueTmp;
378                   }
379                   if(clientPref) writeClientValuePreference(valueWrapper.
                        ID, value);
380                   dialog.dismiss();
381         }
382       });
383       Button cancel = (Button) dialog.findViewById(R.id.cancel);
384       cancel.setOnClickListener(new OnClickListener() {
385         @Override
386         public void onClick(View v) {
387           dialog.dismiss();
388         }
389       });
390
391       dialog.show();
392
```

```
393     } else {
394       // instance of PrefsListItemWrapper, i.e. client log flags
395       PrefsListItemWrapper wrapper = (PrefsListItemWrapper) item;
396       if(Logging.DEBUG) Log.d(Logging.TAG,"PrefsActivity onItemClick
              super " + wrapper.ID);
397
398       dialog.setContentView(R.layout.prefs_layout_dialog_selection);
399       final ArrayList<ClientLogOption> options = new ArrayList<
              ClientLogOption>();
400       String[] array = getResources().getStringArray(R.array.
              prefs_client_log_flags);
401       for(String option: array) options.add(new ClientLogOption(option));
402       ListView lv = (ListView) dialog.findViewById(R.id.selection);
403       new PrefsLogOptionsListAdapter(this, lv, R.id.selection, options);
404
405       // setup buttons
406       Button confirm = (Button) dialog.findViewById(R.id.confirm);
407       confirm.setOnClickListener(new OnClickListener() {
408         @Override
409         public void onClick(View v) {
410           ArrayList<String> selectedOptions = new ArrayList<String>();
411           for(ClientLogOption option: options) if(option.selected)
                  selectedOptions.add(option.name);
412           if(Logging.DEBUG) Log.d(Logging.TAG, selectedOptions.size() + "
                   log flags selected");
413           new SetCcConfigAsync().execute(formatOptionsToCcConfig(
                  selectedOptions));
414            dialog.dismiss();
415         }
416       });
417       Button cancel = (Button) dialog.findViewById(R.id.cancel);
418       cancel.setOnClickListener(new OnClickListener() {
419         @Override
420         public void onClick(View v) {
421           dialog.dismiss();
422         }
423       });
424
425       dialog.show();
426     }
427   }
428
429   @Override
430   protected void onDestroy() {
431       if(Logging.VERBOSE) Log.v(Logging.TAG,"PrefsActivity onDestroy()");
432       super.onDestroy();
433       doUnbindService();
434   }
435
436   private void writeClientValuePreference(int id, double value) {
437     // update preferences
438     switch (id) {
439     case R.string.prefs_disk_max_pct_header:
440       clientPrefs.disk_max_used_pct = value;
441       break;
442     case R.string.prefs_disk_min_free_gb_header:
443       clientPrefs.disk_min_free_gb = value;
444       break;
```

```
445     case R.string.prefs_network_daily_xfer_limit_mb_header:
446       clientPrefs.daily_xfer_limit_mb = value;
447       break;
448     case R.string.battery_charge_min_pct_header:
449       clientPrefs.battery_charge_min_pct = value;
450       break;
451     case R.string.battery_temperature_max_header:
452       clientPrefs.battery_max_temperature = value;
453       break;
454     case R.string.prefs_cpu_number_cpus_header:
455       clientPrefs.max_ncpus_pct = value;
456       //convert value back to number for layout update
457       value = pctCpuCoresToNumber(value);
458       break;
459     case R.string.prefs_cpu_time_max_header:
460       clientPrefs.cpu_usage_limit = value;
461       break;
462     case R.string.prefs_cpu_other_load_suspension_header:
463       clientPrefs.suspend_cpu_usage = value;
464       break;
465     case R.string.prefs_memory_max_idle_header:
466       clientPrefs.ram_max_used_idle_frac = value;
467       break;
468     default:
469       if(Logging.DEBUG) Log.d(Logging.TAG,"onClick (dialog submit button)
              , couldnt match ID");
470       Toast toast = Toast.makeText(getApplicationContext(), "ooops!
              something went wrong...", Toast.LENGTH_SHORT);
471       toast.show();
472       return;
473     }
474     // update list item
475     updateValuePref(id, value);
476     // preferences adapted, write preferences to client
477     new WriteClientPrefsAsync().execute(clientPrefs);
478   }
479
480   private double numberCpuCoresToPct(double ncpus) {
481     double pct = (ncpus / (double)hostinfo.p_ncpus) * 100;
482     if(Logging.DEBUG) Log.d(Logging.TAG,"numberCpuCoresToPct: " + ncpus +
            hostinfo.p_ncpus + pct);
483     return pct;
484   }
485
486   private double pctCpuCoresToNumber(double pct) {
487     double ncpus = (double)hostinfo.p_ncpus * (pct / 100.0);
488     if(ncpus < 1.0) ncpus = 1.0;
489     return ncpus;
490   }
491
492   public Double parseInputValueToDouble(String input) {
493     // parse value
494     Double value = 0.0;
495     try {
496       input=input.replaceAll(",","."); //replace e.g. European decimal
              seperator "," by "."
497       value = Double.parseDouble(input);
498       if(Logging.DEBUG) Log.d(Logging.TAG,"parseInputValueToDouble: " +
```

```
            value);
499         return value;
500       } catch (Exception e) {
501         if(Logging.WARNING) Log.w(Logging.TAG, e);
502         Toast toast = Toast.makeText(getApplicationContext(), "wrong format
                !", Toast.LENGTH_SHORT);
503         toast.show();
504         return null;
505       }
506     }
507
508     private String formatOptionsToCcConfig(ArrayList<String> options) {
509       StringBuilder builder = new StringBuilder();
510       builder.append("<cc_config>\n <log_flags>\n");
511         for(String option: options) builder.append("  <" + option + "/>\n")
                ;
512         builder.append(" </log_flags>\n <options>\n </options>\n</cc_config
                >");
513       return builder.toString();
514     }
515
516     private final class WriteClientPrefsAsync extends AsyncTask<
          GlobalPreferences,Void,Boolean> {
517       @Override
518       protected Boolean doInBackground(GlobalPreferences... params) {
519         if(mIsBound) return monitor.setGlobalPreferences(params[0]);
520         else return false;
521       }
522
523       @Override
524       protected void onPostExecute(Boolean success) {
525         if(Logging.DEBUG) Log.d(Logging.TAG,"WriteClientPrefsAsync returned
                : " + success);
526         updateLayout();
527       }
528     }
529
530     private final class SetCcConfigAsync extends AsyncTask<String,Void,
          Boolean> {
531       @Override
532       protected Boolean doInBackground(String... params) {
533         if(Logging.DEBUG) Log.d(Logging.TAG,"SetCcConfigAsync");
534         if(mIsBound) monitor.setCcConfig(params[0]);
535         else if(Logging.WARNING) Log.w(Logging.TAG,"SetCcConfigAsync
                monitor not bound!");
536         return true;
537       }
538     }
539
540     public class ClientLogOption {
541       public String name;
542       public Boolean selected = false;
543
544       public ClientLogOption(String name) {
545         this.name = name;
546       }
547     }
548 }
```