# Achieving Better Performance Through True Best Effort in Scavenging Grid Computing

## Melhoria de Desempenho em Grades Computacionais Oportunistas

Raphael de Aquino Gomes
Universidade Federal de Goiás
Instituto de Informática
Goiânia-GO, Brazil
raphael@inf.ufg.br

Fábio Moreira Costa
Universidade Federal de Goiás
Instituto de Informática
Goiânia-GO, Brazil
fmc@inf.ufg.br

Fouad Joseph Georges
Universidade Salgado de Oliveira
Núcleo Tecnológico
Goiânia-GO, Brazil
fouadjg@gmail.com

## ABSTRACT

In addition to an untuned performance, inefficient resource management in hinders any attempt to offer Quality of Service in scavenging grids. In this case, Best-Effort mechanisms are synonym to unreliability. Evidently it would be impossible, with undedicated resources, to offer any deterministic warranty (if based on a classic reservation of resources) to service access. Although the usage of scavenging grids for real-time execution remains a challenge, new "Service Qualities" may be proposed to compatiblize applications' preferences versus system oscillations.

## Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications; C.1.4 [**Parallel Architectures**]: Distributed architectures

## General Terms

Performance

## Keywords

Grid computing, Scavenging grids, Adaptive systems, QoS, Performance monitoring

## RESUMO

Além de uma performance desafinada, um gerenciamento pouco eficiente dos recursos coíbe qualquer tentativa de oferecimento de Qualidade de Serviço (QoS), onde o melhor-esforço se torna sinônimo de inconfiabilidade. De fato, já com recursos não dedicados, seria impossível oferecer garantia determinística, baseada na clássica reserva de recursos, no acesso aos serviços. Enquanto a utilização de ambientes de grade oportunistas para execução de aplicações de tempo real, multimídia e tele-imersão, ainda constitui um desafio, novas "Qualidades de Serviço" podem ser propostas às aplicações para compatibilizar suas preferências diante das oscilações do sistema. Esse artigo apresenta uma arquitetura de verdadeiro melhor esforço baseado na comparação do perfil das aplicações e dos nós da grade.

## Palavras-chave

Computação em Grade, Grades Oportunistas, Sistemas Adaptativos, QoS, Monitoramento de Performance



## 1. INTRODUÇÃO

O principal estímulo para o desenvolvimento de grades computacionais oportunistas [15] é a quantidade considerável de recursos que geralmente não é utilizada em estações de trabalho [12]. Ambientes nessa categoria de sistema distribuído (como InteGrade [8], Condor [14] e OurGrid [4]) são essencialmente formados por nós heterogêneos e não-dedicados, oferecendo às aplicações garantias limitadas ao melhor-esforço.

Uma vez que os recursos não são dedicados, não é possível oferecer garantia determinística da Qualidade de Serviço (QoS) no acesso aos serviços, aquela baseada na reserva de recursos. Diante disso, a utilização de ambientes de grade para execução de aplicações de tempo real, multimídia e tele-imersão, ainda constitui um desafio.

A maioria dos trabalhos desenvolvidos visando melhoria de desempenho e QoS em grades oportunistas atuam no escalonamento das aplicações que fazem uso da grade ([3], [2]). Contudo, uma grade oportunista não possui determinismo suficiente para garantir que os recursos ociosos no momento do escalonamento continuarão disponíveis. Apesar da existência de trabalhos que buscam identificar perfis de aplicações visando obter um escalonamento mais preciso ([3], [1]), não é possível obter a garantia mencionada uma vez que os perfis inferidos podem sofrer anomalias e geralmente são baseados em funções aritméticas que escondem detalhes como rajadas de utilização dos recursos. Consequentemente, atuar somente no escalonamento não estabelece os requisitos necessários. É preciso buscar essas variáveis também para aplicações que já foram escalonadas e estão executando.

O desempenho e QoS de aplicações já escalonadas está in-

trisicamente relacionada com aplicações locais. Estas têm prioridade absoluta sobre aquelas, de modo que na sua ocorrência os recursos devem ser liberados, tornando-se indisponíveis. A indisponibilidade de um recurso representaria uma falha em um sistema computacional clássico. Já em um ambiente de grade existe a possibilidade de utilização de recursos adicionais, de modo que a ação mais indicada na ocorrência dessa "falha" seria migrar a aplicação, reescalonando-a para outro nó da grade. Contudo, migrar aplicações na ocorrência de cada falha não é desejável uma vez que migrações frequentes podem afetar a execução de aplicações, além de outros problemas como a possibilidade de não existirem recursos adicionais [13]. É necessário evitar migrações, possivelmente através da tomada de medidas alternativas para contornar o problema de falta de recursos e, consequentemente, melhorar o desempenho e QoS para a aplicação.

Este artigo discute a análise de padrões de aplicações locais como forma de evitar migrações de aplicações da grade e, consequentemente, melhorar o desempenho, através de um verdadeiro melhor-esforço. É proposto o uso de técnicas de adaptação dinâmica como alternativa à migração. O enfoque é na análise de aplicações locais buscando identificar padrões do comportamento destas, diferente de outras abordagens que buscam identificar padrões das aplicações da grade ou de utilização dos recursos como um todo, sem considerar cada aplicação local de forma isolada. Nosso trabalho é desenvolvido sobre o middleware de grade InteGrade [8].

O restante do artigo é organizado da seguinte forma: na seção 2 são discutidos trabalhos relacionados; na seção 3 é fornecida uma visão geral do InteGrade; a seção 4 apresenta detalhes da nossa proposta, ao passo que o modelo de avaliação desta é discutido na seção 5; a seção 6 apresenta algumas conclusões e discute trabalhos futuros.

## 2. TRABALHOS RELACIONADOS

Grande parte dos trabalhos que visam incluir mecanismos de QoS e melhorar o desempenho em grades oportunistas atuam nas operações que são executadas antes das aplicações iniciarem sua execução, mais especificamente nos algoritmos e políticas de escalonamento. Dentre estes, [3] analisa a influência da heterogeneidade dos nós da grade sobre o desempenho das aplicações, criando algoritmos de escalonamento que levam em consideração as propriedades do nó voluntário, como disponibilidade e volatilidade. Esse trabalho se assemelha ao nosso uma vez que busca identificar perfis de utilização dos nós da grade, mas atua especificamente no escalonamento, e o perfil buscado é em relação ao nó como um todo e não para cada aplicação como pretendemos.

Soluções clássicas para QoS, como reserva de recursos, são adotadas em [22] e [19]. O primeiro analisa o impacto da reserva de recursos, projetando algoritmos de escalonamento que levam em consideração essa informação. Já [19] alia reserva a controle de admissão, estaticamente caracterizando a demanda de recursos e criando perfis das aplicações de grade. Os resultados desses trabalhos os classificam como boas soluções para balancear as necessidades de aplicações remotas e locais mas não garantem um bom desempenho na ocorrência de uma aplicação local cujas necessidades podem invalidar a reserva. Nosso objetivo é melhorar o desempenho buscando evitar deficiências como essa.

Em [13] são oferecidas garantias estatísticas de QoS para aplicações de tempo real numa grade oportunista. São definidas duas métricas: **taxa de falha de tarefa** – probabilidade de uma tarefa falhar devido a falta de recursos ociosos e **probabilidade de migrações ruins** – probabilidade de uma tarefa ser migrada imediatamente após uma migração anterior. Com base em estatísticas e definição de perfil dos recursos disponíveis [16], são estabelecidas heurísticas que determinam o número máximo de tarefas que podem ser admitidas pelo sistema e um período de tempo que uma tarefa deve esperar para ocupar um recurso ocioso, de forma que as métricas se mantenham abaixo de valores especificados. Nosso tratamento busca traçar um perfil de utilização mais preciso, permitindo a execução de um número maior de aplicações simultaneamente pois não realizaremos controle de admissão.

No *Grid Harvest Service* (GHS) [21] é proposto um arcabouço para escalonamento de aplicações paralelas que utiliza predição ao nível do sistema e das aplicações. Da mesma forma que pretendemos, o sistema é monitorado durante a execução das tarefas visando identificar possíveis anomalias aos padrões identificados na predição. Contudo, a única saída tomada para aumentar o desempenho é atuar no escalonamento. Pretendemos alcançar resultados mais satisfatórios realizando uma análise mais ampla não nos limitando apenas aos dados da predição na análise e tomando medidas alternativas como adaptação.

## 3. VISÃO GERAL DO INTEGRADE

Nosso trabalho é desenvolvido sobre a plataforma de middleware de grade InteGrade[1] [8]. Essa plataforma foi escolhida por suportar diversas categorias de aplicações paralelas, entre as quais se destaca o modelo ***Bulk Synchronous Parallelism*** (BSP) [17], além de aplicações seqüenciais e paramétricas. Outro ponto levado em consideração foi o fato de que essa plataforma não leva em consideração QoS e desempenho para aplicações da grade, sendo um de seus principais requisitos a garantia dessas características apenas para os usuários que compartilham seus recursos com a grade.

Desenvolvido como iniciativa do IME-USP (Instituto de Matemática e Estatística da Universidade de São Paulo) em convênio com outras universidades e grupos de pesquisa, o InteGrade possui uma arquitetura orientada a objetos, sendo construído sobre o padrão CORBA [18] de objetos distribuídos.

Uma grade InteGrade é composta por um conjunto de aglomerados, organizados em uma rede hierárquica, onde cada aglomerado é composto por uma coleção de máquinas localizadas numa rede local. Cada aglomerado possui um nó responsável por gerenciar o aglomerado, chamado **Nó de Gerenciamento**, e diversos nós que compartilham recursos com a grade, denominados **Nós Provedores de Recursos**, além do **Nó de Usuário**, a partir do qual um usuário pode submeter aplicações para execução na grade. Estas categorias para os nós não são exclusivas, por exemplo podemos ter

[1] http://www.integrade.org.br

uma máquina que é nó provedor de recursos e nó de usuário simultaneamente [6].

A Figura 1 ilustra a disposição dos módulos que compõem o InteGrade. O LRM (*Local Resource Manager*) coleta informações sobre o nó em que é executado. O GRM (*Global Resource Manager*) é um módulo com a funcionalidade de escalonar as tarefas aos outros nós da grade através das informações enviadas pelos LRMs. O NCC (*Node Control Center*) é usado pelo usuário da máquina para controlar as políticas de acesso a seus recursos. O ASCT (*Application Submission and Control Tool*) é utilizado pelo usuário para submeter aplicações a serem executadas na grade, o que também pode ser feito através do Portal, via web. O LUPA (*Local Usage Pattern Analyzer*) monitora os padrões de uso, enquanto o GUPA (*Global Usage Pattern Analyzer*) auxilia o GRM no escalonamento através das informações enviadas pelos LUPAs. O AR (*Application Repository*) armazena as aplicações a serem executadas. A biblioteca BSPlib [9] permite que aplicações C/C++ escritas para a implementação de Oxford da BSPlib [10] sejam executadas no InteGrade.

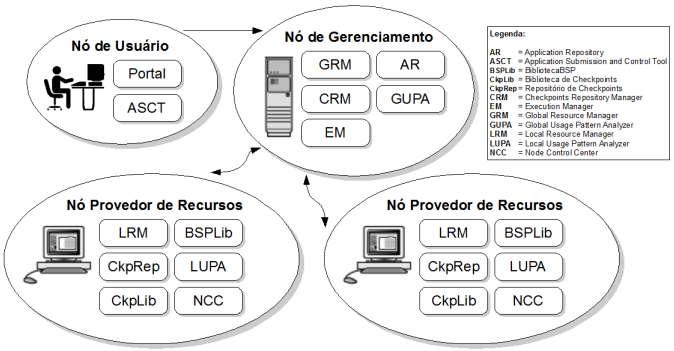


**Figure 1: Arquitetura de um aglomerado do InteGrade.**

Os outros componentes da arquitetura são responsáveis pelas operações de recuperação por retrocesso. A CkpLib (Biblioteca de *checkpointing*) é responsável por gerar *checkpoints* contendo o estado de um processo para posterior recuperação, feita por funções da mesma biblioteca. O CkpRep (Repositório de *checkpoints*) armazena *checkpoints* e arquivos de saída gerados por processos executando na grade. O EM (*Execution Manager*) é responsável por manter uma lista com as aplicações em execução no aglomerado, incluindo a localização de cada processo da aplicação e o estado da requisição de execução. O CRM (*Checkpoints Repository Manager*) mantém informações sobre os repositórios de *checkpoints* presentes em seu aglomerado, incluindo seu endereço de rede e lista de *checkpoints* armazenados.

O fluxo de submissão e execução de aplicações no InteGrade, em um cenário em que não ocorre falhas é descrito a seguir:

1. O usuário registra sua aplicação no AR através do ASCT ou do Portal e solicita sua execução utilizando os mesmos módulos. O usuário pode, opcionalmente, especificar requisitos para a execução de sua aplicação, como por exemplo a quantidade mínima de memória necessária para a execução.
2. Assim que a requisição é recebida, o GRM procura um nó candidato para executar a aplicação com base nos requisitos da aplicação informados pelo usuário, a disponibilidade de recursos na grade, além do padrão de uso inferido pelo LUPA. Caso nenhum nó satisfaça os requisitos da aplicação ou mesmo nos casos em que não há nós ociosos, o GRM notifica tal fato ao ASCT finalizando o protocolo. Entretanto, caso haja algum nó que satisfaça os requisitos, o GRM envia a solicitação para o LRM da máquina candidata e o estado da requisição ao EM.
3. O LRM solicita a aplicação ao AR e os eventuais arquivos de entrada ao ASCT requisitante, e lança a aplicação, notificando ao ASCT que sua requisição foi atendida, informando seu endereço ao EM.
4. Durante a execução são gerados *checkpoints* do estado da aplicação que são armazenados nos repositórios de *checkpoints* através do CRM.
5. Caso não ocorra nenhuma falha[2] e a aplicação seja executada com sucesso, o LRM atualiza o estado da solicitação no EM. No caso em que os recursos são solicitados, o LRM informa esse fato ao EM que executa o procedimento de reinicialização da aplicação em outro nó.

# 4. PROJETO DO MECANISMO DE MELHORIA DE DESEMPENHO

O escalonamento padrão do InteGrade consiste num mecanismo simples que se baseia apenas nos requisitos fornecidos na submissão da aplicação. Buscando maximizar o desempenho do sistema, o escalonamento foi otimizado de forma que, de posse das informações colhidas pelo LUPA e da demanda da aplicação, é feita a melhor divisão de tarefa/nó possível. Nessa otimização não é estabelecido nenhum compromisso (contrato) entre o sistema e a aplicação, não tendo o usuário nenhuma garantia com relação ao desempenho obtido.

Uma informação que poderia ser levada em consideração no processo de escalonamento é o fato de que aplicações distintas geralmente possuem interesses distintos e, consequentemente, requisitos distintos. Mas mesmo com essa otimização, atuar apenas no escalonamento não garante o desempenho [13]. Isso se dá em parte devido ao fato de que os módulos LUPA e GUPA possuem algumas limitações, como o fato de se basearem no comportamento padrão de uso dos recursos, não capturando comportamentos anômalos (que fogem ao padrão). Outra carência encontrada é o fato de que informações sobre picos (rajadas) de utilização dos recursos são eliminadas pois esses módulos consideram apenas a média de utilização.

Para suprir as carências identificadas acima é preciso lidar com alguns problemas:

1. Determinar o grau de persistência da situação anômala ou rajada em um nó.

[2]Na versão atual do InteGrade a solicitação de recursos por aplicações locais no nó provedor de recursos, mesmo que passageira, configura uma falha.

2. Determinar um novo nó para onde relocar a aplicação prejudicada, no caso de migração.
3. Estimar a duração:
    - da aplicação da grade em execução (e que está sendo prejudicada);
    - da anomalia ou rajada (que provavelmente depende da duração de algum programa local lançado pelo usuário);
    - da migração da tarefa para outro nó (tempo de relocação somado ao tempo de recuperação).

O processo de migração possui custos como alocação de recursos e transporte de dados que, em alguns casos, podem prejudicar a aplicação, tornando esse processo não-justificável. Assim sendo, é preciso determinar se é viável fazer a migração ou não. Uma primeira abordagem é procurar identificar o que causou a anomalia ou rajada e, a partir da estimativa do tempo de sua duração, comparar com o tempo necessário para retomar a execução em outro nó.

Caso a migração tenha um custo alto, ou mesmo em casos em que não existem recursos disponíveis para realizá-la, devem ser adotadas saídas alternativas para contornar o problema de perda de desempenho. Nossa proposta é utilizar mecanismos de adaptação que permitam a continuidade da execução da aplicação. Esses mecanismos podem ser tanto executados localmente, atuando sobre o nó onde a aplicação está executando, quanto sobre a estrutura do funcionamento da grade, como modificando o algoritmo de escalonamento para que futuras alocações de recursos não causem a "falha" identificada.

## 4.1 Arquitetura

A Figura 2 ilustra os módulos a serem incluídos ao InteGrade e sua relação com aqueles já existentes, paralelamente ao ciclo de ações que podem ser realizadas durante a execução da aplicação e a fonte das informações fornecidas como entrada.

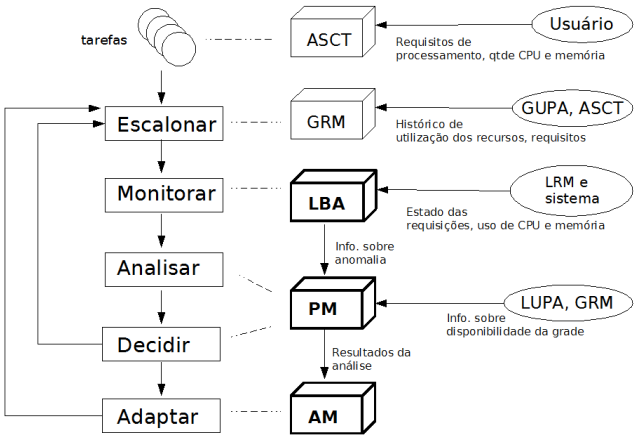


**Figure 2: Proposta de trabalho. Módulos em destaque serão incorporados ao InteGrade.**

- ***Local user pattern anomaly and Burst Analiser* (LBA):** Módulo responsável por analisar rajadas de utilização dos recursos buscando identificar comportamentos anômalos ou consumo que prejudique o funcionamento de aplicações críticas. Esse módulo será uma extensão do LUPA, possivelmente atuando com auxílio deste.
- ***Performance Manager* (PM):** Módulo responsável por, com base nos dados referentes à utilização do recurso, decidir se a tarefa deve ser migrada ou se algum mecanismo adaptativo deve ser acionado.
- ***Adaptation Manager* (AM):** Módulo responsável por implementar os mecanismos adaptativos descritos anteriormente. Esses mecanismos podem atuar no nó onde a tarefa está executando como forma de melhorar o desempenho sem migração ou podem atuar nos outros módulos da grade, como os responsáveis pelo escalonamento. Uma primeira abordagem é realizar essas adaptações mediante Reflexão Computacional [11].

O protocolo de execução de aplicações no InteGrade com a inclusão desses novos módulos seria uma extensão do protocolo discutido na seção 3, diferindo de forma mais significativa nos casos em que é identificada uma "falha" por falta de recursos. As ações incluídas são discutidas abaixo:

1. Após a submissão da aplicação e início de sua execução, o LBA passaria a monitorar o estado da aplicação pelo LRM e a utilização dos recursos, visando identificar níveis que impeçam a execução da aplicação. Esses níveis podem se tratar de rajadas esporádicas que não são identificados pelo LUPA; ou comportamentos anômalos não detectados no padrão inferido, ou seja, um nível de utilização que normalmente não ocorre naquele nó.
2. Ao detectar utilização do recurso acima de um certo nível, que perdure por um período superior ao suportável, o LBA aciona o PM para que as medidas adequadas sejam tomadas.
3. O PM analisa os dados passados pelo LBA buscando identificar, com o auxílio das informações fornecidas pelo LUPA, se a utilização dos recursos é algo temporário, configurando uma rajada[3] ou se se trata de uma utilização que persistirá por um período considerável (anomalia). Em ambos os casos, o PM compara o tempo de duração da rajada e/ou anomalia com o tempo necessário para migrar a aplicação para outro nó utilizando o mecanismo de *checkponting*. Essa comparação é feita visando identificar a melhor saída entre migração e adaptação. Informações adicionais, como características dos nós disponíveis para migração, além do que está atualmente sendo utilizado, podem ser considerados na comparação.
4. Após tomar a decisão com base nas informações disponíveis, é feita uma de duas operações:

    4.1. o EM é acionado para que a aplicação seja migrada para outro nó;

[3]Mesmo nos casos em que for identificada uma rajada é preciso analisar com que frequência essa rajada pode ocorrer, uma vez que algumas categorias de aplicação podem ser prejudicadas pela ocorrência de rajadas frequentes.

4.2. são realizados mecanismos adaptativos pelo AM de forma a permitir a continuação da execução da aplicação no nó atual. Como alternativa ou mesmo como operação adicional, a adaptação pode ser feita em outros componentes da grade visando evitar que o problema encontrado ocorra novamente, como adaptação do GRM modificando o escalonamento. Nesse segundo caso continua ocorrendo migração da aplicação mas com probabilidade menor de falha.

Nosso projeto está sendo desenvolvido no Instituto de Informática (INF) da Universidade Federal de Goiás (UFG) encontrando-se na fase de projeto detalhado dos módulos. Ainda não existe uma implementação dos módulos que permita realizar uma avaliação precisa dos prós e contras da nossa abordagem. Contudo, experimentos já realizados demonstram a existência do problema. Esses experimentos são discutidos na próxima seção juntamente com uma descrição dos experimentos que pretendemos realizar para comprovar a eficácia da nossa proposta.

# 5. AVALIAÇÃO

Na seção anterior descrevemos nossa abordagem para melhoria de desempenho e QoS para aplicações em grades oportunistas. Nós propomos a análise de padrões de aplicações locais como forma de prever o uso dos recursos e, com isso, evitar migrações desnecessárias. Também propomos o uso de técnicas de adaptação dinâmica como alternativa à migração.

Nesta seção discutimos alguns experimentos que nos motivaram a desenvolver essa pesquisa, assim como estudos experimentais que realizaremos para avaliar a eficiência de nossas técnicas.

Com nossos experimentos pretendemos responder as seguintes questões:

- A ocorrência de rajadas e/ou anomalias ocorre em um nível que pode atrapalhar o desempenho de aplicações? Em que nível isso ocorre?
- É possível prever ou estimar o tempo de ocorrência de uma certa rajada ou anomalia, assim como a quantidade de recurso que é utilizada em virtude dessa?
- Nossa arquitetura consegue obter uma melhor utilização dos recursos da grade?
- Nossa arquitetura melhora o desempenho da grade e, consequentemente, a qualidade de serviço das aplicações que dela fazem uso?

Através de experimentos já foram obtidas respostas parciais a algumas dessas perguntas. Pretendemos responder as demais perguntas com novos experimentos.

## 5.1 Experimentos realizados

Coletamos a utilização de CPU e memória em uma máquina em intervalos de 1 segundo, buscando analisar o real uso desses recursos. Os dados de CPU foram coletados diretamente do `/proc/stat` e os da memória utilizando o comando `free` no sistema operacional Slackware Linux [20].

Os experimentos foram realizados em um DELL Optiplex 745, Intel Core 2 Duo 2.2GHz, 4 GB de RAM. Por ser considerada uma estação de trabalho com grande capacidade de processamento, acredita-se que os resultados obtidos sejam ainda mais críticos em estações menos "poderosas". Outra característica relevante é o fato dessa estação fazer parte de um laboratório de ensino, o que a caracteriza como ideal para uma grade oportunista, pois se encontra a maior parte do tempo ociosa.

Como sugerimos, existem rajadas de utilização de recursos que não seriam coletados pelo LUPA. Um exemplo encontrado através de nossos experimentos é apresentado na Figura 3. Essa figura ilustra um cenário típico que poderia ocorrer. Durante 1 hora (13:00:00 às 14:00:00) a utilização média da CPU na estação analisada ficou entre 2,98%, com um desvio padrão de 7,98%. Na implementação atual, o LUPA colhe a média de utilização de 5 em 5 minutos. Assim, se o fizesse entre 13:39:00 e 13:44:00 obteria 8,43% como nível médio de utilização. Contudo, de 13:39:58 a 13:40:38 (40 segundos) a utilização ficou acima de 50%, o que poderia ser crucial para causar o fracasso de aplicações que requerem 50% ou mais como nível mínimo de ociosidade.

Como qualquer padrão inferido está sujeito a possíveis anomalias [7] não é necessário, e nem seria justificável, realizar experimentos para comprovar a existência destas. A anomalia é algo que pode surgir independente do recurso utilizado e apenas não ocorreria se supormos que os recursos utilizados são dedicados. Mas considerando que uma grade oportunista é formada quase sempre por nós compartilhados, essa suposição não é verdadeira.

## 5.2 Projeto da Avaliação

Nesta seção são descritos alguns experimentos que pretendemos realizar para validar nosso trabalho e comprovar que o monitoramento e análise de anomalias/rajadas aliada a técnicas de adaptação podem melhorar o desempenho das aplicações da grade. Esses experimentos serão realizados após o término da implementação para avaliação, consolidação e homologação de nossa abordagem.

Os experimentos são descritos a seguir mas provavelmente não nos limitaremos a estes pois novas necessidades poderão surgir ao longo do desenvolvimento do projeto.

- **Sobrecarga causada pelos módulos nos nós compartilhados**

  Uma das principais preocupações numa grade oportunista é que os módulos da grade não interfiram no desempenho do usuário local, consumindo o mínimo de recursos possível. Dessa forma pretendemos avaliar a sobrecarga de nossos módulos nos nós compartilhados, buscando analisar se é viável sua execução. Isso é importante para manter o nível de sobrecarga baixa: os módulos já existentes no InteGrade não consomem muitos recursos nos nós compartilhados [8, 5].

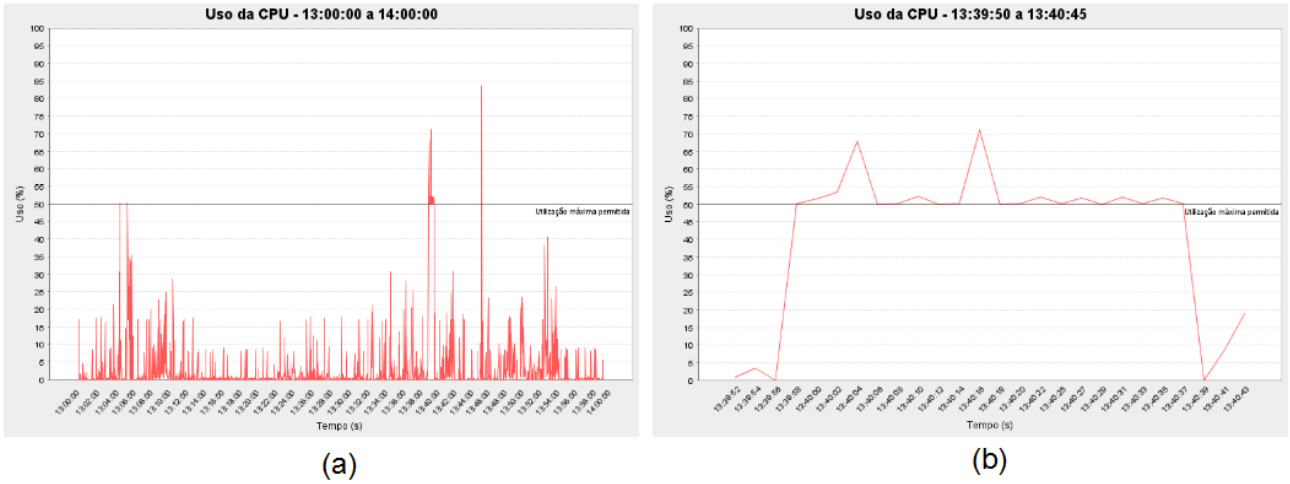


**Figure 3: (a) Utilização dos recursos durante uma hora do dia. (b) Rajada de utilização nesse período.**

- **Taxa de falhas causadas por requisição dos recursos**

  A taxa de falhas das aplicações da grade devido a falta de recursos será o principal alvo de nossos experimentos pois isso representa o fator mais crítico a ser analisado. Mediremos esse valor na implementação corrente do InteGrade, sem a inclusão de nossos módulos e com a inclusão desses. Esperamos que os valores obtidos no primeiro caso sejam reduzidos satisfatoriamente.

- **Tempo de finalização das aplicações da grade**

  Como a principal variável levada em consideração para avaliar o desempenho e, em alguns casos, a QoS é o tempo de finalização da aplicação, pretendemos avaliar esse valor em nosso sistema. Analisaremos o tempo de finalização das tarefas com a inclusão de nossos módulos e também sem a utilização desses visando realizar um estudo comparativo. Analisaremos também se a granularidade da grade (quantidade de nós participantes) influencia no desempenho das aplicações e em que proporção isso ocorre.

# 6. CONCLUSÕES E TRABALHOS FUTUROS

O middleware InteGrade atualmente não possui tratamentos visando a garantia de qualidade de serviço para as aplicações da grade. Isto significa que, ao enfrentar a concorrência de aplicações locais em determinados nós da grade, as aplicações de grade devem ser desalocadas e re-escalonadas, possivelmente a partir de um *checkpoint* armazenado. No presente trabalho, nos propomos a investigar uma forma mais eficiente de recuperação da aplicação, tanto em termos de ganhos de desempenho quanto em relação a questões de sincronismo, concorrência e consistência entre as tarefas da aplicação.

Uma outra contribuição deste trabalho é a inclusão de mecanismos adaptativos no InteGrade, os quais permitirão a adaptação dinâmica dos serviços da grade visando melhorias em seu desempenho global, principalmente em termos da vazão de execução de aplicações. A adaptação dinâmica será também considerada como alternativa ao tratamento baseado na migração de tarefas. Desta forma, investigaremos o uso de técnicas adaptativas que permitirão melhorar a qualidade de serviço da aplicação sem a necessidade de migrar suas tarefas.

Com isto, esperamos contribuir para tornar viável o uso de grades oportunistas por parte de aplicações com requisitos de qualidade de serviço, tais como aplicações de tempo-real (multimídia distribuída, por exemplo).

O trabalho encontra-se atualmente em andamento, sendo que a definição da arquitetura geral encontra-se consolidada. As próximas etapas consistirão na implementação e avaliação das técnicas propostas, procurando demonstrar e quantificar seu uso e benefícios.

# 7. AGRADECIMENTOS

Esse projeto recebe apoio financeiro do CNPq, Brasil (processo n. 557396/2008-5).

# 8. REFERÊNCIAS

[1] G. C. Bezerra. Análise de Conglomerados Aplicada ao Reconhecimento de Padrões de Uso de Recursos Computacionais. Master's thesis, Department of Computer Science - University of São Paulo, São Paulo, March 2006.

[2] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman. Heuristics for scheduling parameter sweep applications in gridenvironments. *Heterogeneous Computing Workshop, 2000.(HCW 2000) Proceedings. 9th*, pages 349–363, 2000.

[3] S. Choi. *Group-based Adaptive Scheduling Mechanism in Desktop Grid*. PhD thesis, Department of Computer Science and Engineering Graduate School – Korea University, June 2007.

[4] W. Cirne, F. Brasileiro, N. Andrade, L. Costa, A. Andrade, R. Novaes, and M. Mowbray. Labs of the World, Unite!!! *Journal of Grid Computing*, 4(3):225–246, 2006.

[5] D. M. R. Conde. Análise de padrões de uso em grades

computacionais. Master's thesis, Department of Computer Science - University of São Paulo, São Paulo, January 2008.

[6] R. Y. de Camargo. *Armazenamento distribuído de dados e checkpointing de aplicações paralelas em grades oportunistas*. Dr.scient. afhandling, nstituto de Matemática e Estatística – Universidade de São Paulo, 2007.

[7] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley-Interscience, 2000.

[8] A. Goldchleger. Fabio Kon an Alfredo Goldman, Marcelo Finger, and Germano Capistrano Bezerra. Integrade: Object-oriented grid middleware leveraging idle computing power of desktop machines. *Concurrency and Computation: Practice & Experience*, 16:449–459, 2004.

[9] A. Goldchleger, C. Queiroz, F. Kon, and A. Goldman. Running Highly-Coupled Parallel Applications in a Computational Grid. *Proceedings of the 22th Brazilian Symposium on Computer Networks (SBRC 2004)(Gramado-RS, Brazil, May 2004)*.

[10] J. Hill, B. McColl, D. Stefanescu, M. Goudreau, K. Lang, S. Rao, T. Suel, T. Tsantilas, and R. Bisseling. BSPlib: The BSP programming library. *Parallel Computing*, 24(14):1947–1980, 1998.

[11] F. Kon, F. Costa, G. Blair, and R. H. Campbell. The case for reflective middleware. *Commun. ACM*, 45(6):33–38, 2002.

[12] D. Kondo, M. Taufer, C. Brooks, H. Casanova, and A. Chien. Characterizing and evaluating desktop grids: an empirical study. *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, 2004.

[13] J. Liang and M. Nahrstedt. Supporting quality of service in a non-dedicated opportunistic environment. *Cluster Computing and the Grid, 2004. CCGrid 2004. IEEE International Symposium on*, pages 74–81, 2004.

[14] M. Litzkow, M. Livny, and M. Mutka. Condor-a hunter of idle workstations. *Distributed Computing Systems, 1988., 8th International Conference on*, pages 104–111, 1988.

[15] M. Livny, J. Basney, R. Raman, and T. Tannenbaum. Mechanisms for high throughput computing. *SPEEDUP Journal*, 11(1):36–40, 1997.

[16] M. Mutka and M. Livny. The available capacity of a privately owned workstation environment. *Performance Evaluation*, 12(4):269–284, 1991.

[17] M. Norman. Bulk Synchronous Parallelism. *WoTUG Newsletter*, 17:32–35, 1992.

[18] Object Management Group. CORBA v3.0 Specification. *OMG Document 02-06-33*, July 2002.

[19] J. Rolia, X. Zhu, M. Arlitt, and A. Andrzejak. Statistical service assurances for applications in utility grid environments. *Performance Evaluation*, 58(2-3):319–339, 2004.

[20] R. SunOS. Slackware Linux, OSF 3.2. *BSDi, NetBSD, OpenBSD and FreeBSD are not*.

[21] M. Wu and X. Sun. A general self-adaptive task scheduling system for non-dedicated heterogeneous computing. *Cluster Computing, 2003. Proceedings. 2003 IEEE International Conference on*, pages 354–361, 2003.

[22] M. Wu, X. Sun, and Y. Chen. QoS Oriented Resource Reservation in Shared Environments. *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)-Volume 00*, pages 601–608, 2006.