Análise de desempenho de algoritmos de escalonamento em simuladores de grades

Alvaro Henry Mamani Aliaga
alvaroma@ime.usp.br

Departamento de Ciência da Computação
IME-USP

Dezembro de 2009

Conteúdo

Universidade de São Paulo

Contexto

- Computação em grade
- Escalonadores
- Algoritmo de escalonamento
- Simulação

Universidade de São Paulo

Computação em grade

- Computação em grade (*Grid Computing*)
- É uma alternativa para obter grande capacidade de processamento.

Figura: Fonte: http://www.adarshpatil.com/newsite/grid.htm

Universidade de São Paulo

Algoritmos de escalonamento

- Para alcançar o potencial de recursos distribuídos, efetivos e eficientes algoritmos de escalonamento são de fundamental importância.
- O problema de encontrar um escalonamento ótimo é NP-difícil.
- Num ambiente de computação em grade, cada recurso é dinâmico
- Avaliação dos algoritmos para problemas seqüênciais.

Universidade de São Paulo

Algoritmos de escalonamento

Dynamic FPLTF

- FPLTF (*Fastest Processor to Largest Task First*)
- FPLTF possui uma versão dinâmica: **Dynamic FPLTF**
- *Dynamic FPLTF* precisa de três tipos de informação para escalonar:
  - Tamanho da tarefa.
  - Velocidade do host.
  - Carga do host.
- O *Time to Become Available* (TBA) de cada host é inicializado com 0, e as tarefas são ordenadas pelo tamanho.
- Esta estratégia tenta minimizar os efeitos da dinâmicidade da grade.

Universidade de São Paulo

Algoritmos de escalonamento

Sufferage

- Uma máquina é atribuída a uma tarefa que poderia ser o "*suffer*".
- O problema deste algoritmo é o mesmo problema do *Dynamic FPLTF*; esto precisa de muita informação para calcular a *completion time* das tarefas. É necessário conhecer *TASK SIZE*, *HOST LOAD* e *HOST SPEED*

Universidade de São Paulo

Algoritmos de escalonamento

Algoritmo Workqueue

- é uma escalonamento de conhecimento livre
- Tarefas são escolhidas em uma ordem arbitrária e enviadas ao processador.
- A idéia atrás disto é que mais tarefas serão atribuídas para máquinas rápidas/ociosas enquanto que máquinas lentas/ocupadas processarão uma pequena carga. A vantagem aqui é que *Workqueue* não depende de informação de desempenho.

Universidade de São Paulo

Algoritmos de escalonamento

Algoritmo WorkQueue with Replication

- Foi criado para solucionar problemas que não precisam de informação dos recursos da grade para fazer o escalonamento.
- As tarefas são enviadas para execução nos processadores que se encontram disponíveis em uma ordem aleatória.
- Quando um processador finaliza a execução de uma tarefa, este recebe uma nova tarefa para processar.
- As heurísticas WQR e *Workqueue* passam a diferir no momento em que um processador se torna disponível e não há mais nenhuma tarefa pendente para executar.

Universidade de São Paulo

Algoritmos de escalonamento

Algoritmo XSufferage

- O algoritmo *XSufferage*, usa informações sobre os níveis, os quais precisam estar disponíveis no momento em que o algoritmo vai alocar as tarefas. Algumas das informações que ele precisa são:
  - disponibilidade de CPU;
  - disponibilidade da rede;
  - Os tempos de execução das tarefas.
- *XSufferage* é uma extensão de heurística de escalonamento *Sufferage*.

Universidade de São Paulo

Algoritmos de escalonamento

Algoritmo Storage Affinity

- Leva em conta o fato dos dados de entrada da aplicação serem freqüentemente reutilizados em execuções sucessivas.
- Um registro da localização dos dados utilizados pela aplicação é mantido e permite que o escalonador associe as tarefas de forma a evitar, tanto quanto possível, transferências desnecessárias de grandes quantidades de dados.
- Além disso, o efeito de decisões ineficientes de escalonamento é reduzido através do uso de replicação de tarefas

Universidade de São Paulo

Simuladores

- Aplicações reais devem rodar aplicações por longos períodos de tempo, e não é viável para executar um grande número de experimentos tipo simulação neles.
- A utilização de recursos reais, torna difícil a tarefa de explorar uma grande variedade de configurações de recursos.
- Variações na carga de recursos ao longo do tempo tornam difícil a obtenção de resultados reproduzíveis.

Simuladores - SimGrid

Componentes

Como descrito na seguinte figura, a ferramenta SimGrid possui três camadas principais.

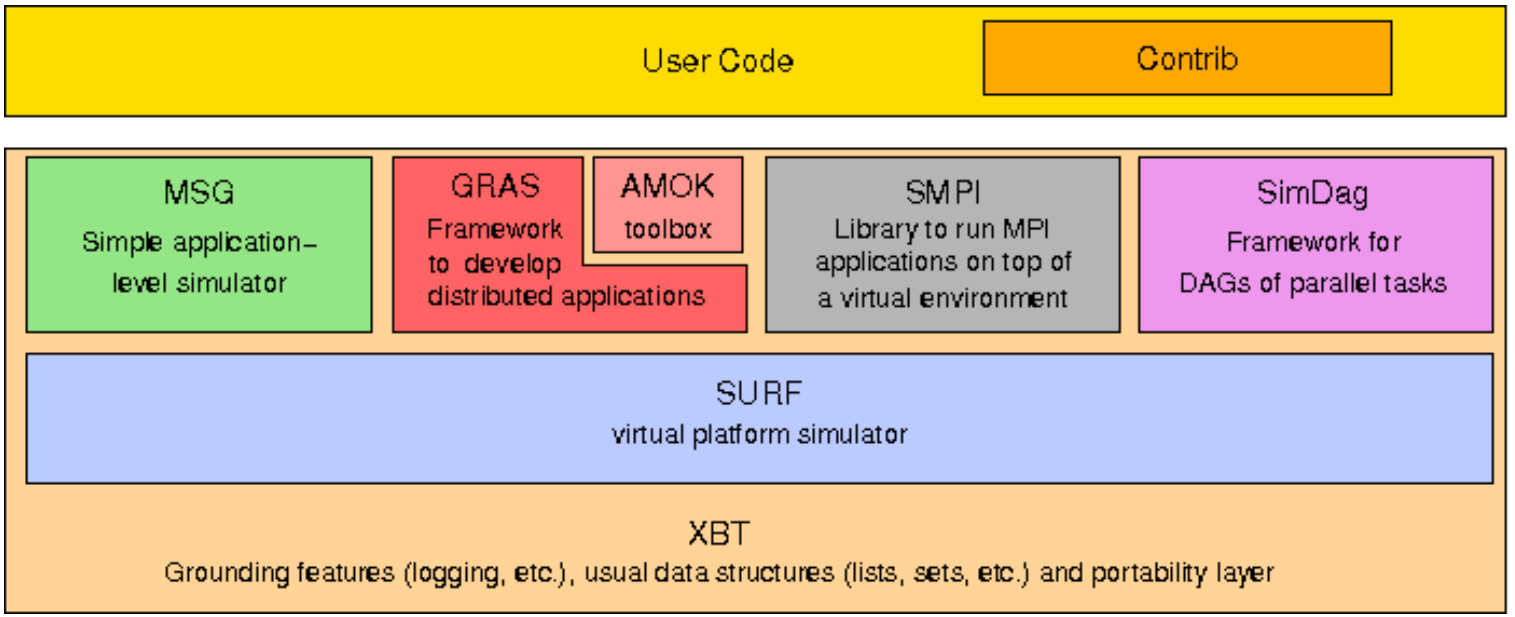


Figura: Módulos do *SimGrid*

Simuladores - SimGrid

Camada: Ambientes de Programação

- **MSG**: É usado para modelar aplicações como processos seqüenciais concorrentes(*Concurrent Sequential Processes*). Útil para modelar problemas teóricos e para comparar diferentes heurísticas.
- **SMPI**: (*Simulated MPI*), para simulações de códigos MPI.
- **GRAS**: (*Grid Reality And Simulation*).
- **SimDag**: ambiente dedicado à simulação de aplicações paralelas, por meio do modelo DAG(*Direct Acyclic Graphs*).

Universidade de São Paulo

Simuladores - SimGrid

Camada: Núcleo de simulação e Base

Núcleo de simulação

- *SURF*: núcleo das funcionalidades.

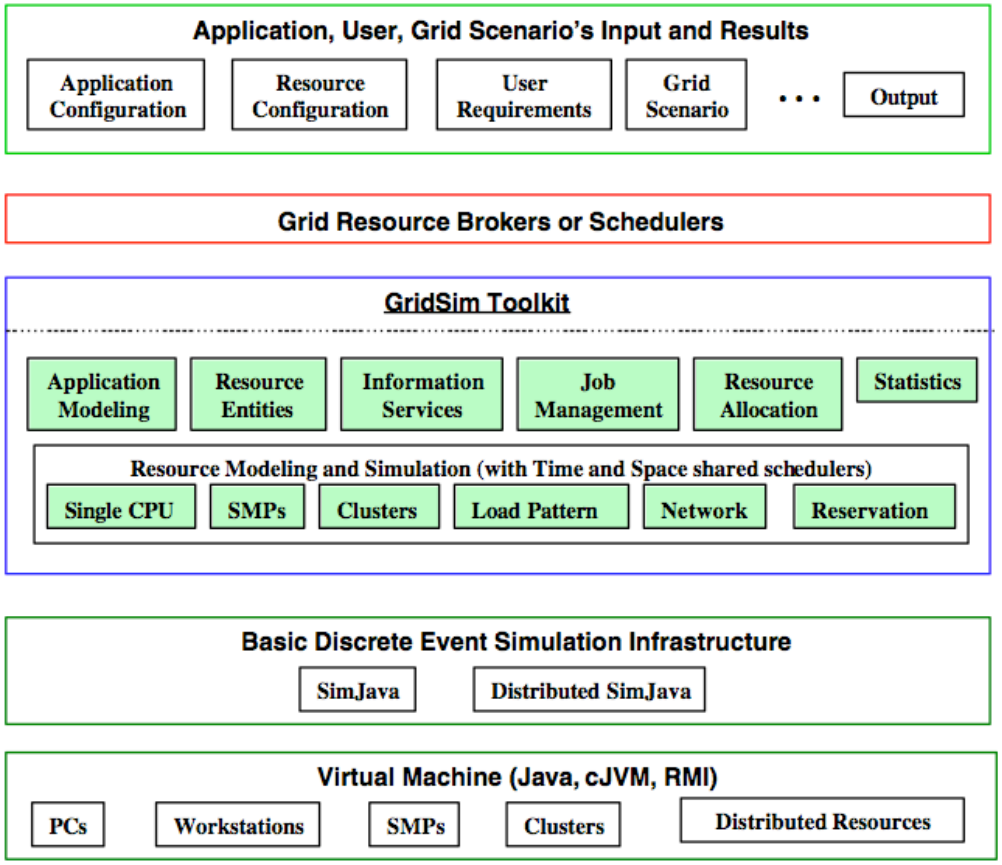
Base

- XBT(*eXtended Bundle of Tools*): provê suporte à portabilidade da ferramenta.

Simuladores - GridSim

Componentes

A arquitetura do simulador foi concebida de forma modular e em camadas. Esta arquitetura multi-camadas e suas aplicações são mostradas na seguinte figura.



Universidade de São Paulo

- *SimJava*, é um pacote de propósito geral para simulação discreta orientada a eventos, implementada em *Java*.
- O comportamento de uma entidade é codificada em *Java* usando seu método *body()*

Simuladores - GridSim

Entidades

- **User**: Cada instância da entidade *User* representa um usuário da grade.
- **Broker**: Cada usuário é conectado a uma instância de uma entidade *Broker*.
- **Recource**: Cada instância desta entidade representa um recursos da grade.
- **Grid Information Service**: Provê o serviço de registros dos recursos existentes na grade.
- **Input e Output**: O fluxo de informações entre as entidades do *GridSim*.

Universidade de São Paulo

Resultados da literatura

Do trabalho: Heuristic for Scheduling Parameter Sweep Applications in Grid Environments

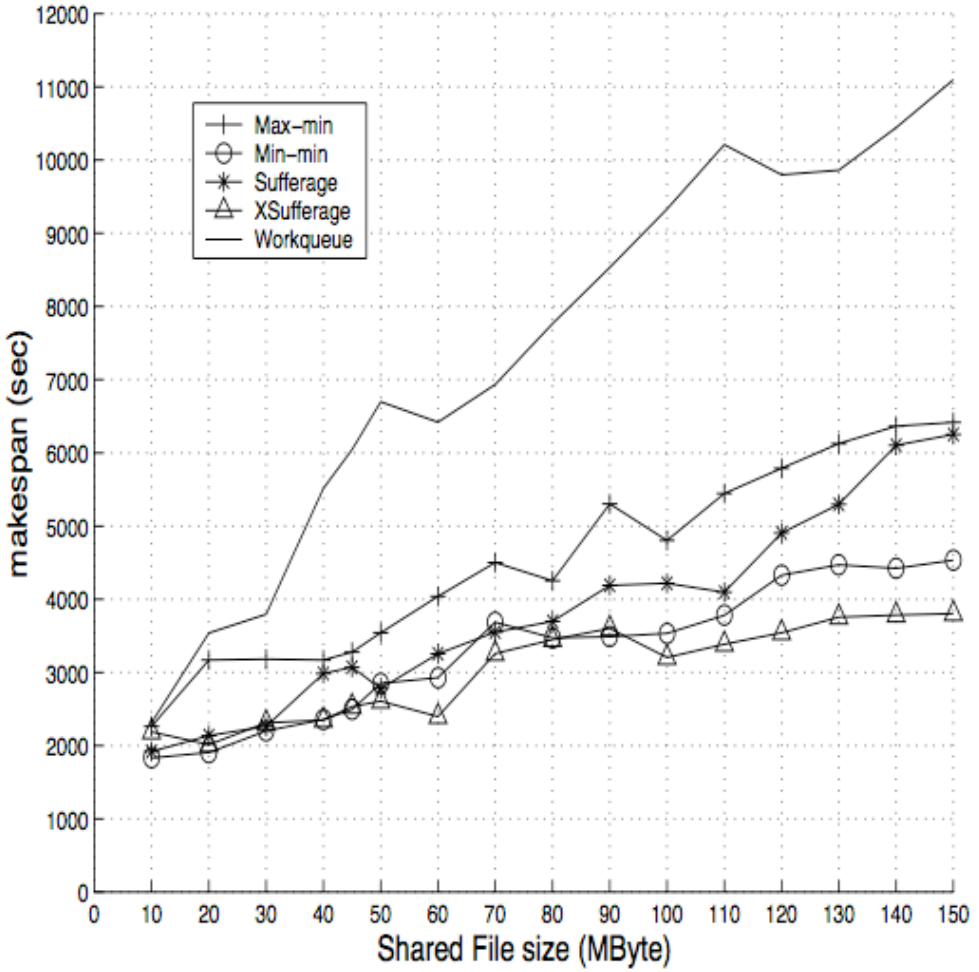


Figura: Comparação dos algoritmos

Universidade de São Paulo

Resultados da literatura

Do trabalho: Trading Cycles for Information: Using Replication to Schedule Bag-of-Task Applications on Computacional Grids

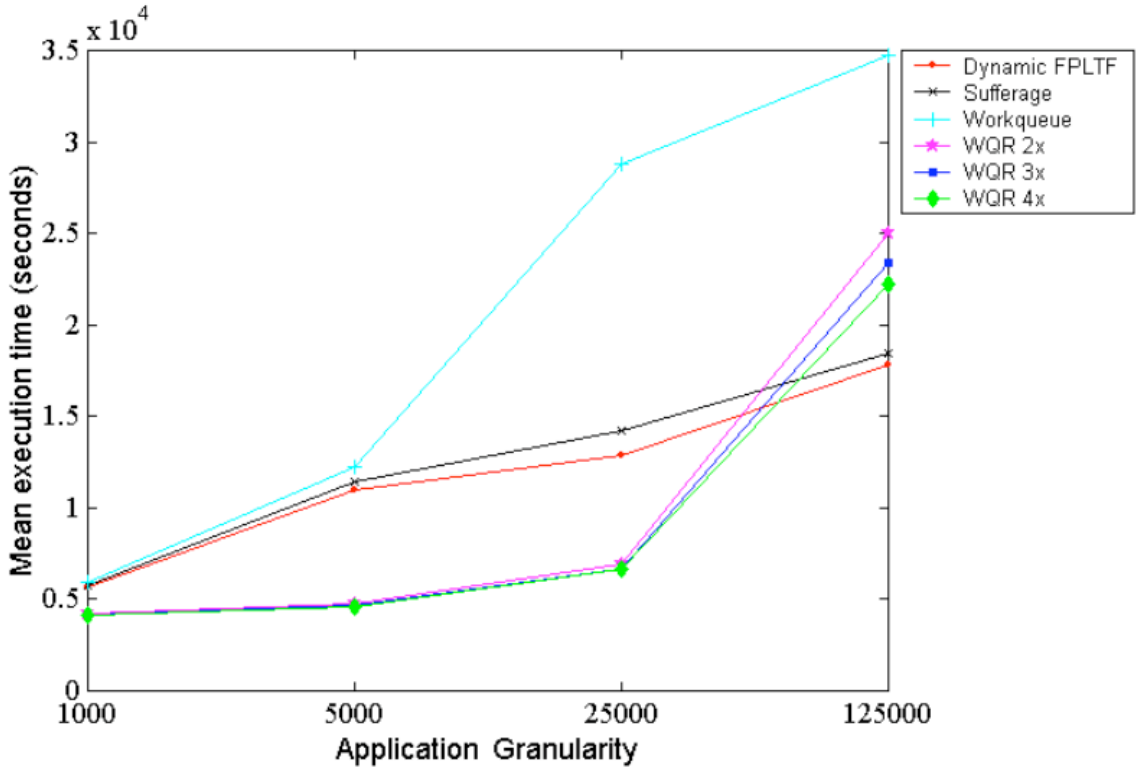


Figura: Comparação dos algoritmos

Universidade de São Paulo

Resultados da literatura

Do trabalho: Trading Cycles for Information: Using Replication to Schedule Bag-of-Task Applications on Computacional Grids

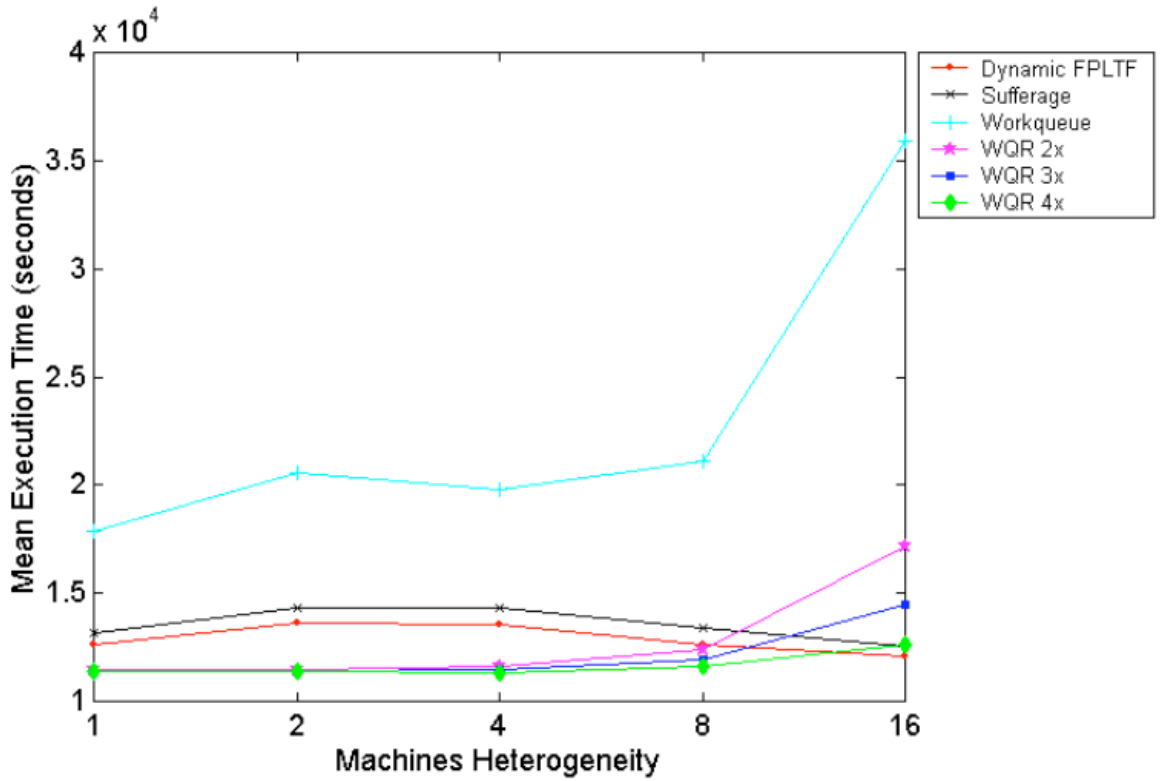


Figura: Comparação dos algoritmos

Universidade de São Paulo

Resultados da literatura

Do trabalho: Trading Cycles for Information: Using Replication to Schedule Bag-of-Task Applications on Computacional Grids

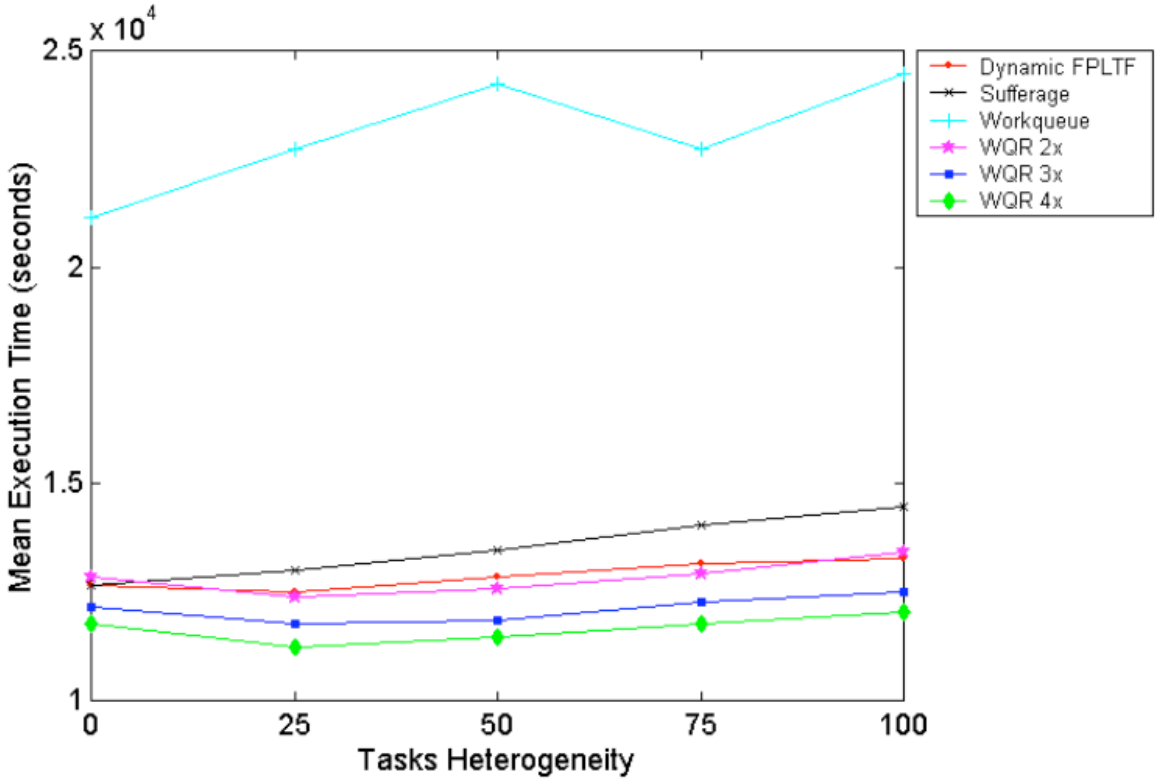


Figura: Comparação dos algoritmos

Universidade de São Paulo

Resultados da literatura

Do trabalho: Trading Cycles for Information: Using Replication to Schedule Bag-of-Task Applications on Computacional Grids

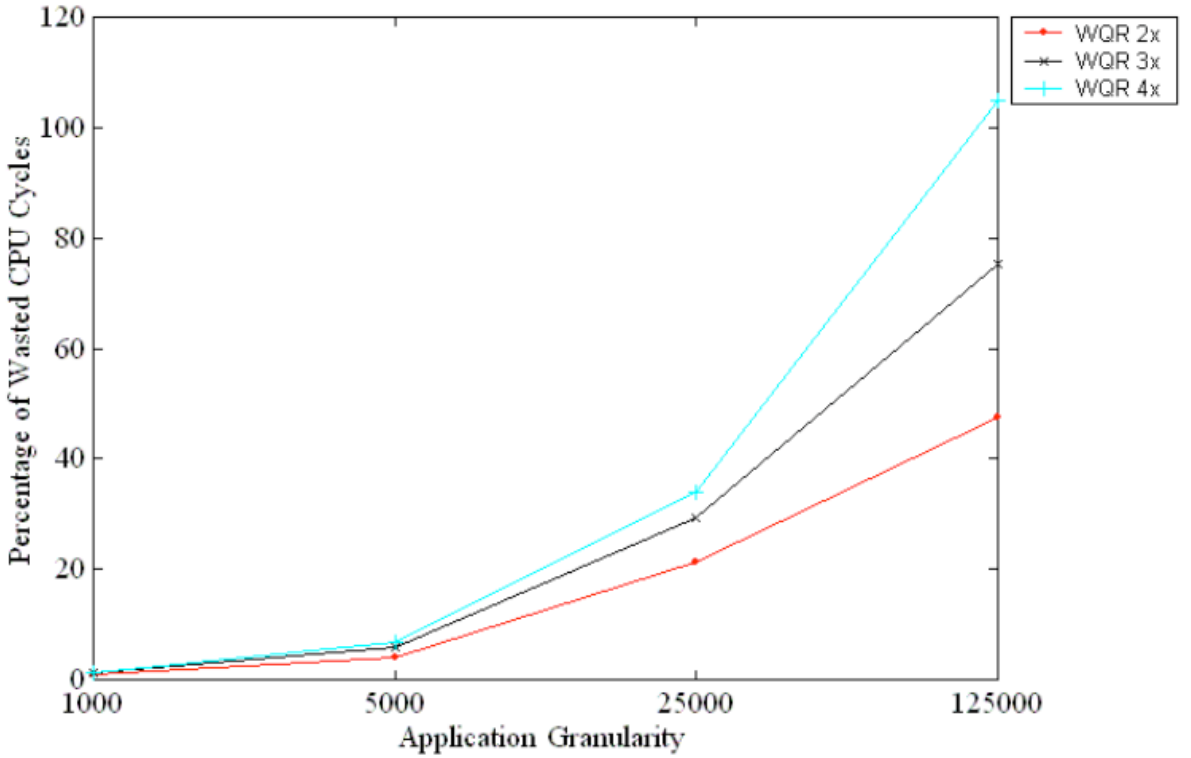


Figura: Comparação dos algoritmos

Universidade de São Paulo

Resultados da literatura

Do trabalho: Avaliação de algoritmos de escalonamento para tarefas em Grids

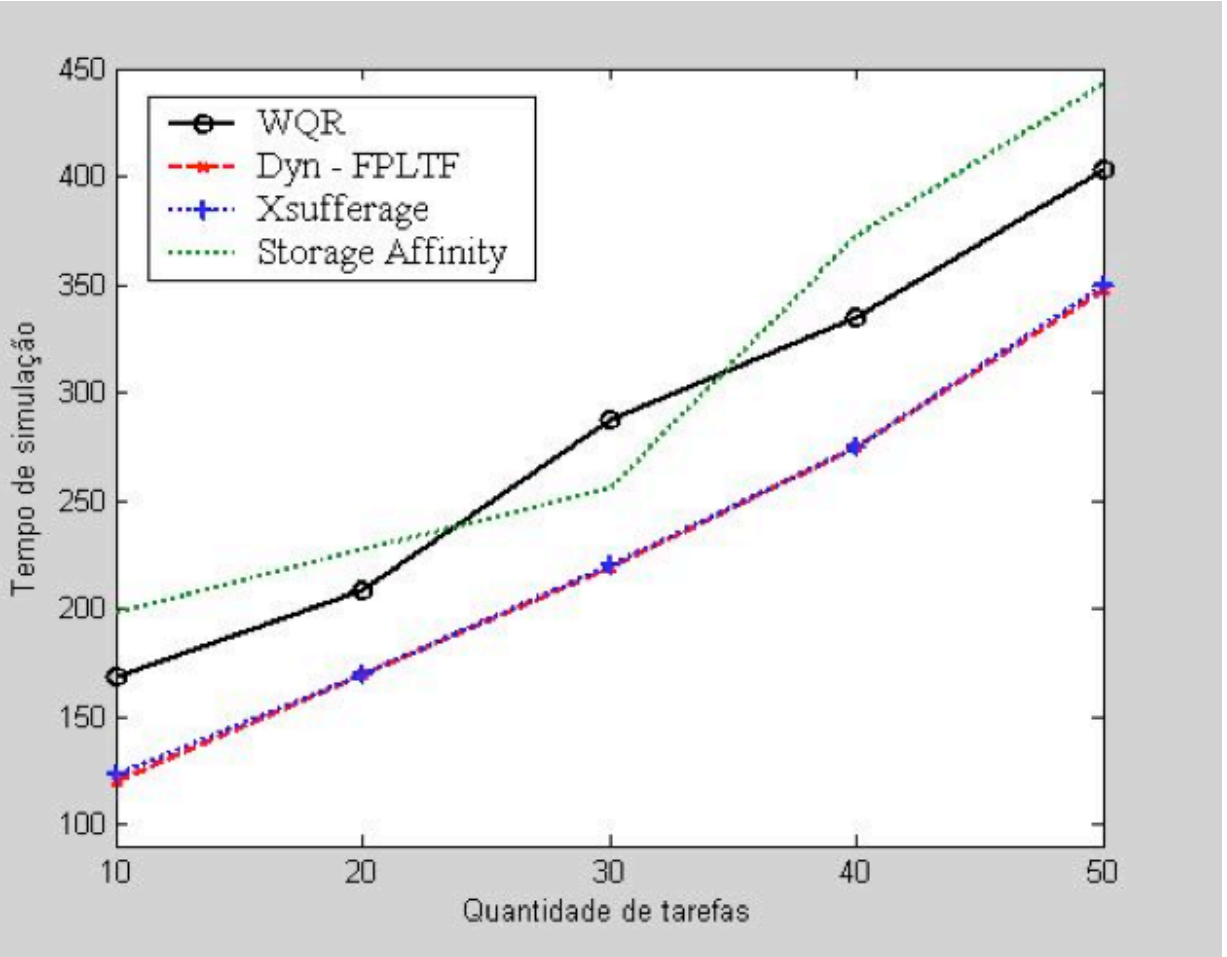


Figura: Comparação dos algoritmos

Universidade de São Paulo

Conclusões e trabalhos futuros

- Neste estudo foram avaliados os principais algoritmos para escalonamento de tarefas em Grids computacionais. Na literatura foram avaliados esses algoritmos principalmente com o simulador SimGrid. Principalmente, os algoritmos WQR e Storage Affinity têm seu dinâmismo caracterizado pela criação e distribução de replicas. O algoritmo Dynamic FPLTF analisa a carga de processamento, para que tarefas grandes não sejam alocadas a máquinas lentas. XSufferage não é dinâmico.
- Simulação tomando diferentes características da grade.
- Simulações em tipos de aplicações paralelas.

Universidade de São Paulo

Obrigado!!!

Universidade de São Paulo