Análise de Desempenho Quantitativo do Sistema de Gerenciamento de Tarefas OpenPBS sobre um aglomerado de PCs

Martha Torres[1] , Edgar Charry[2]

[1]Colegiado de Ciência da Computação - DCET – Universidade Estadual de Santa Cruz (UESC)
Rod. Ilhéus/Itabuna, Km 16 CEP 45.650-000 Ilhéus - BA – Brasil

[2]Engenharia Elétrica – Escola Politécnica – Universidade de São Paulo (USP)
Av. Luciano Gualberto, tr. 3 no. 380 - Cidade Universitária CEP 05508-900 São Paulo - SP – Brasil
mxtd@vision.ime.usp.br, echarry@hotmail.com

Abstract. *This paper presents the quantitative performance evaluation of the job management system OpenPBS on a aglomerados of PCs. The methodology used is based in some published methods. The OpenPBS performance in function of task size (with regard to processors number), task execution time and different load conditions is analyzed using dynamic workloads. The OpenPBS overhead and the scheduling algorithm behavior are showed using synthetic workloads.*

Resumo. *Este artigo apresenta a avaliação de desempenho quantitativa do sistema de gerenciamento de tarefas OpenPBS sobre uma máquina paralela baseada em aglomerados de PCs. A metodologia adotada para este fim é baseada em diferentes métodos descritos na literatura. Utilizando cargas de trabalho ("workloads") dinâmicas analisa-se o desempenho do OpenPBS em função do tamanho das tarefas (com respeito ao numero de processadores que utilizam), do tempo de execução das tarefas e de diferente tipos de carga. Utilizando cargas de trabalho sintéticas mostra-se o overhead do OpenPBS e também ilustra-se o funcionamento do algoritmo de escalonamento do OpenPBS.*

1. Introdução

Atualmente os aglomerados de computadores estão sendo muito usados como plataformas distribuídas e/ou paralelas, para computação de alto desempenho e/ou para alta disponibilidade. Junto com eles também se tem desenvolvido os sistemas de gerenciamento de tarefas, cujo objetivo é gerenciar da melhor maneira possível os recursos disponíveis dos aglomerados.

Nos sistemas de gerenciamento de tarefas, os usuários submetem suas tarefas em modo "batch" a uma fila do sistema de gerenciamento de tarefas e um escalonador central decide como alocar os recursos para a execução das tarefas. Dependendo da estratégia do sistema de escalonamento, o sistema de gerenciamento de tarefas pode minimizar ou não o tempo de resposta das tarefas.

A motivação principal para realizar esta pesquisa foi a necessidade de conhecer de maneira quantitativa o desempenho do OpenPBS (http://ww.openpbs.org/about.html)

já que é o sistema de gerenciamento de tarefas utilizado em nossa plataforma, além disso por ser de domínio público é um sistema utilizado em muitos outros aglomerados e até hoje nós não temos conhecimento de análises de desempenho quantitativo deste sistema.

Na literatura existem alguns estudos de avaliação quantitativa de sistemas de gerenciamento de tarefas [Gaj et al. 2002][Frachtenberg et al., 2003] e de avaliação de desempenho de algoritmos de escalonamento de tarefas [Sherwani et al., 2002] [Aida 2000][Kettimuthu et al. 2002][Ernemann et al, 2002].

Portanto este artigo tem como objetivo pesquisar as diferentes metodologias existentes, escolher a metodologia apropriada, desenvolver as ferramentas necessárias para realizar os experimentos e realizar uma avaliação quantitativa do comportamento do OpenPBS.

Este artigo é organizado da seguinte maneira. Na seção 2 serão mencionados alguns trabalhos relacionados com a avaliação de desempenho de sistemas de escalonamento de tarefas. Na seção 3 será descrita em detalhe a metodologia adotada para realizar a avaliação de desempenho do OpenPBS. Na seção 4 mostram-se os resultados obtidos e na Seção 5 apresentam-se as conclusões e os trabalho futuros.

2. Trabalhos Relacionados

Os trabalhos realizados sobre avaliação de desempenho de algoritmos de escalonamento de tarefas, basicamente se diferenciam pelas características das cargas de trabalho utilizadas na análise, alguns se baseiam em cargas de trabalho reais como [Ernemann et al. 2002][Kettimuthu et al. 2002], outros geram cargas de trabalho utilizando modelos matemáticos simples [Gaj et al. 2002][Sherwani et al. 2002] e outros utilizam modelos para gerar cargas de trabalho mais realísticas como [Aida 2000][Frachtenberg et al. 2003].

Existem principalmente dois trabalhos diretamente relacionados com os objetivos da presente pesquisa:

O trabalho de [Gaj et al, 2002] que apresenta a avaliação de desempenho quantitativa de vários sistemas de escalonamento de tarefas, sua plataforma escolhida é um sistema "grid" de nove PCs, as cargas de trabalho utilizadas para realizar a avaliação de desempenho estão compostas somente de tarefas seqüenciais. Ele basicamente utiliza três classes de cargas de trabalho, a primeira classe corresponde a tarefas com tempo de execução curto, a segunda classe pertence a tarefas com tempo de execução médio e a última classe corresponde a tarefas com tempo de execução longo. Como eles mesmos escrevem em seu artigo é o primeiro estudo empírico de sistemas de gerenciamento de tarefas publicado. A metodologia descrita no artigo é útil principalmente na comparação de desempenho com outros sistemas de gerenciamento de tarefas. Mas as cargas de trabalho consideradas, não refletem um comportamento real além de somente considerar tarefas seqüenciais na análise.

O trabalho de [Frachtenber et al, 2003] é um excelente trabalho sobre avaliação quantitativa de sistemas de gerenciamento de tarefas. Ele utiliza o sistema de gerenciamento de recursos STORM [Frachtenber et al, 2000] e sua plataforma é um aglomerado de 32 PCs. Entre seus objetivos estão analisar o efeito do multiprocessamento e do incremento de carga no desempenho do sistema de

gerenciamento de tarefas. Neste trabalho adota-se um modelo desenvolvido por Lublin [Dublin et al. 2003] para gerar a carga de trabalho utilizada e as medidas de desempenho apresentadas são calculadas de maneira geral. Parte de nossa análise de desempenho também é baseada neste modelo, mas nossas medidas de desempenho são mais especializadas permitindo realizar conclusões mais elaboradas**.**

3. Metodologia

O objetivo deste artigo é realizar uma análise de desempenho quantitativo do sistema de gerenciamento de tarefas: OpenPBS.

Uma consideração fundamental é escolher o tipo de carga de trabalho que será submetido para o OpenPBS. É muito importante usar uma carga de trabalho representativa que tenha as mesmas características como uma carga de trabalho real. Um ponto importante das cargas de trabalho reais é que elas são dinâmicas, ou seja, elas chegam em qualquer tempo e o seu tempo de execução também não pode ser previsto. Além disso, o número de tarefas ativas muda com o tempo, em um período de tempo o sistema pode estar vazio enquanto que em outro pode estar lotado.

Uma maneira de obter cargas de trabalho dinâmicas é usar modelos de simulação. Um bom modelo de simulação é aquele que emprega procedimentos estatísticos rigorosos. Que se baseia em "logs" de mais de duas localizações diferentes. E o mais importante que modele paralelismo, tempos de execução, a relação entre eles e também o processo de chegada de cada tarefa [Aida 2000][Ernemann et al. 2002][Frachtenberg et al. 2003]. Para o estudo realizado neste trabalho, foi adotado o modelo de carga de trabalho proposto em [Lublin et al. 2003] que possui as características anteriormente mencionadas e sua implementação está disponível publicamente em http://www.cs.huji.ac.il/labs/parallel/ workload/models.html.

Uma segunda consideração importante é escolher medidas de desempenho representativas para realizar a análise. De maneira geral, as medidas de desempenho representativas mais utilizadas para avaliar o comportamento de um sistema de escalonamento de tarefas são "Turnaround Time" médio e "Slowdown" [Frachtenberg et al. 2003] [Ernemann et al. 2002][Aida 2000][Gaj et al. 2002][Kettimuthu et al. 2002]

Portanto neste trabalho as medidas utilizadas são: o "Turnaround Time" médio que está definido como:

$$TTM = \left(\sum_{i=1}^{i=n} Tespera_i + Trun_i\right) \Big/ n$$

Onde $Tespera_i$ é o tempo que cada tarefa fica na fila do sistema de escalonamento de tarefas e $Trun_i$ é o tempo de execução propriamente dito de cada tarefa, n é o número de tarefas.

E o "Slowdown" médio que está definido como:

$$SM = \left(\sum_{i=1}^{i=n} (Tespera_i + Trun_i)\right) \Big/ \left(\sum_{i=1}^{i=n} Trun_i\right)$$

Para os experimentos realizados neste trabalho foram utilizadas cargas de trabalho de 1000 tarefas que vão chegando em um período de oito dias. Com o objetivo de reduzir o tempo de resposta de nossos experimentos foi adotada a seguinte estratégia [Frachtenberg et al. 2003]: tanto os tempos de chegada quanto os tempos de rodada

foram divididos por uma constante: 100; assim os oito dias puderam ser emulados em aproximadamente 2 horas. Depois de obter a carga de trabalho com as 1000 tarefas e "encurtar" seu tempo de execução é necessário escolher o tipo de programas que vão fazer parte da carga de trabalho.

Neste caso foi escolhido um programa que gera combinações de números em sua versão paralela e seqüencial [Torres et al. 2003]. Razões fundamentais para esta seleção é que estes programas somente gastam tempo de processamento, pouco de comunicação e pouco de memória e principalmente pode-se ter um total controle sobre o seu tempo de execução.

A plataforma utilizada para realizar os experimentos é o aglomerado Biowulf/IME (http://www.vision.ime.usp.br/~cage/Beowulf) que neste momento possui oito nós de processamento (AMD), um nó de gerenciamento (AMD) e um switch FastEthernet de 100 Mbits/s. Cada nó de processamento possui um processador Athlon K7 de 1.2 Ghz, 768 MB de SDRAM. O sistema operacional é o Linux Debian woody com kernel 2.4.24, usa a biblioteca de passagem de mensagens MPICH 1.2.4 e o sistema de gerenciamento de tarefas OpenPBS.

Foram desenvolvidos vários "scripts" em linguagem PERL com o objetivo de ajudar na execução dos experimentos e na sua análise:

O processo de submissão para o OpenPBS foi totalmente automatizado, com a ajuda de um conjunto de "scripts" que recebe a carga de trabalho gerada pelo modelo e seleciona os programas que melhor executam esta carga de trabalho.

Como todo sistema de gerenciamento de tarefas, o OpenPBS produz arquivos "logs" que contem informação de cada tarefa (tempo de chegada, tempo inicio de execução e tempo final de execução), portanto foram criados vários "scripts" para obter as medidas de desempenho necessárias para fazer a análise.

4. Resultados

Inicialmente foi gerada a primeira carga de trabalho para ser executada no OpenPBS, ela está composta de mil tarefas e tempos de chegada de aproximadamente duas horas. Esta é chamada de "carga1" ("load1") e a Figura 1 ilustra a distribuição das tarefas ao longo do tempo em função de seu tempo de execução.

Da figura observa-se que a maioria das tarefas possui tempos de execução pequenos e outras poucas apresentam tempos de execução maiores. Também se pode observar como a chegada de cada tarefa acontece de maneira imprevista tal como se fosse uma carga de trabalho real.

A Figura 2 mostra a distribuição das tarefas ao longo do tempo em função do número de processadores que são requeridos. Como se pode observar, as tarefas com diferente numero de processadores estão bem espalhadas ao longo do tempo. Também se observa que existe maior número de tarefas demandando até quatro processadores depois cai e volta a subir um pouco para oito processadores. Isto reflete um comportamento de cargas de trabalho reais nos quais existem porcentagens altas de tarefas com numero de processadores potencia de dois [Ernemann et al. 2002][Aida 2000][Lublin et al. 2003].

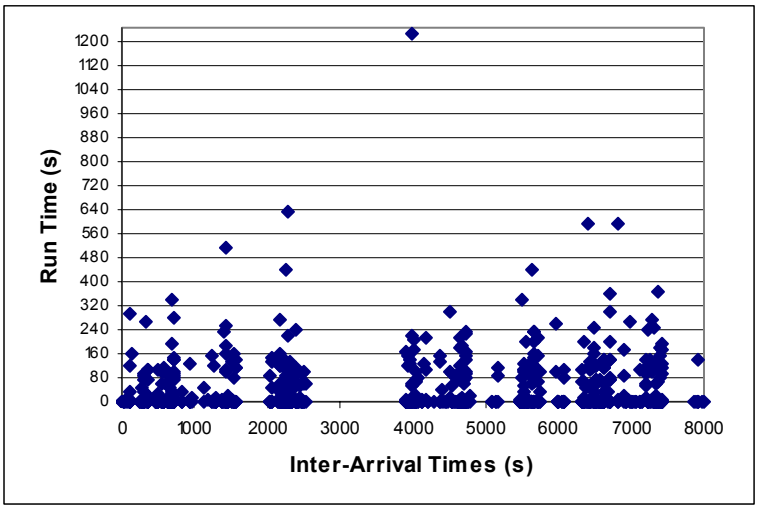

Figura 1. Distribuição das tarefas ao longo do tempo em função de seu tempo de execução.

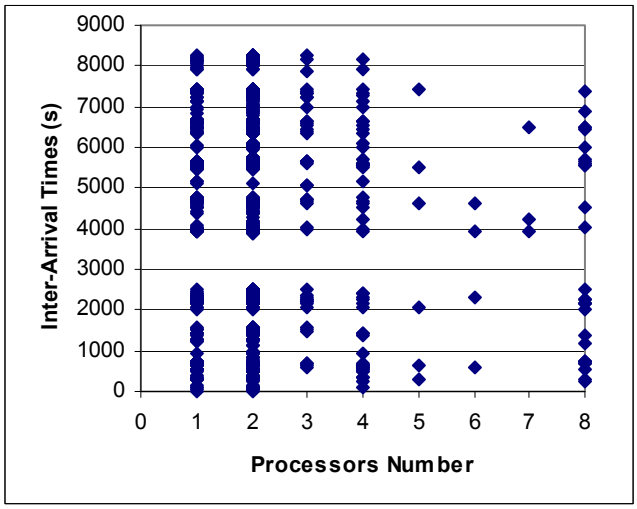

Figura 2. Distribuição das tarefas ao longo do tempo em função do numero de processadores requeridos.

A Figura 3 mostra o tempo de execução das tarefas em função da demanda de número de processadores, esta figura ilustra que os maiores tempos de execução pertencem às tarefas que demandam até três processadores.

Estas três figuras caracterizam de maneira completa a natureza da carga submetida para análise de desempenho do OpenPBS.

Qualquer análise que é baseado nas medidas "Tornaround Time" e "Slowdown" sobre o sistema todo perde informação sobre o comportamento de características especiais das tarefas. Portanto para realizar uma melhor análise dos resultados, as tarefas são classificadas em grupos de acordo com seu tempo de execução e com o número de processadores que demandam e assim as medidas são analisadas para cada grupo. De acordo com o seu tempo de execução têm-se quatro grupos de tarefas: tarefas "muito curtas" ("very short") com tempos de execução ≤ 1 segundo, tarefas "curtas" ("short") com tempos de execução entre 1 e 10 segundos, tarefas "longas" ("long") com

tempos de execução entre 10 e 60 segundos e tarefas “muito longas” (“very long”) com tempos de execução maiores que 60 segundos.

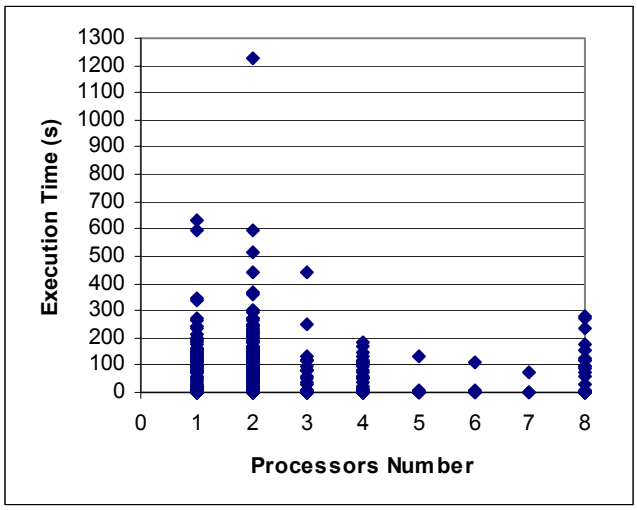

Figura 3. Tempo de execução das tarefas em função do numero de processadores requeridos.

A Figura 4 mostra o “Turnaround Time” médio para tarefas “muito curtas”, neste caso a medida de “Turnaround Time” foi feita com relação ao número de processadores que a tarefa demanda. A Figura 5 mostra a medida de “Slowdown” que proporciona informação sobre a lentidão do sistema. As figuras exibem medidas para tarefas seqüenciais e tarefas de dois até cinco processadores (proc2-5), não existem tarefas menores que 1 segundo que precisem de seis até oito processadores (proc6-8) por isso sua medida não aparece nas figuras.

Este primeiro resultado ilustra que o sistema de escalonamento OpenPBS gera uma sobrecarga de tempo considerável quando as tarefas são “muito curtas”. Estas figuras também ilustram que o OpenPBS deu prioridade para as tarefas seqüenciais apresentando um melhor desempenho para este tipo de tarefas.

A Figura 6 mostra o “Tornaround Time” médio para tarefas “curtas”, para complementar as medidas, a Figura 7 mostra a medida “Slowdown” para o mesmo caso. As figuras ilustram que o OpenPBS apresenta novamente uma sobrecarga de tempo considerável para tarefas “curtas” e também mostra que da prioridade à execução de tarefas seqüenciais e tarefas que demandam menor número de processadores.

As figuras 8 e 9 mostram o “Turnaround Time” médio e “Slowdown” respectivamente para tarefas “longas”. Embora as figuras continuem mostrando que o OpenPBS apresenta uma sobrecarga de tempo alto, pode-se observar que existe um aumento na eficiência do sistema ou seja ele começa a melhorar o tempo de resposta quando as tarefas começam a incrementar seu tempo de execução.

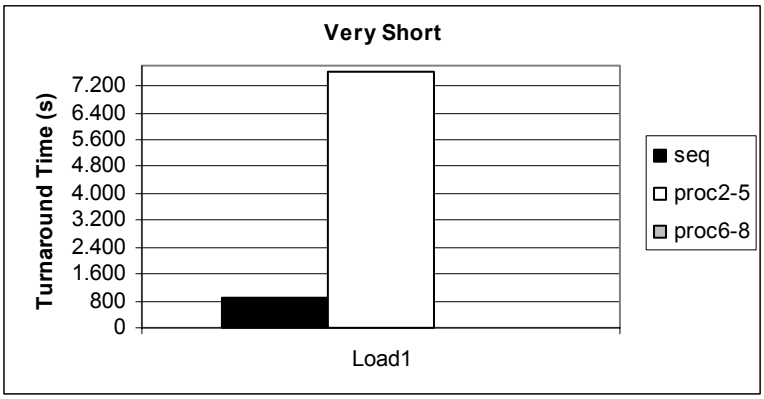

Figura 4. "Turnaround Time" médio para tarefas seqüenciais e paralelas com tempos de execução menores que 1 segundo.

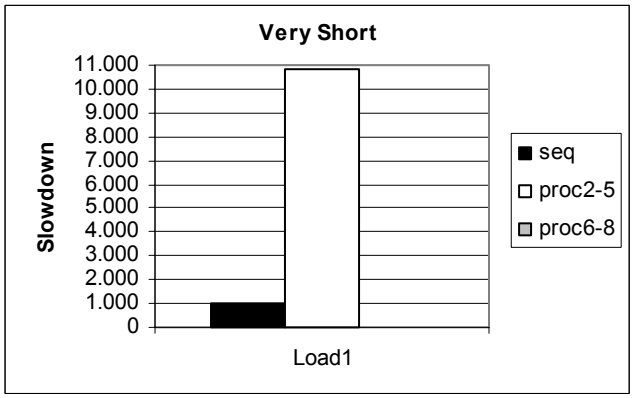

Figura 5. "Slowdown" para tarefas seqüenciais e paralelas com tempos de execução menores que 1 segundo.

As figuras 10 e 11 mostram o "Tornaround Time" médio e "Slowdown" respectivamente para tarefas "muito longas". A sobrecarga de tempo continua sendo muito alta, mas o tempo de espera pela execução das tarefas é mais reduzido quando comparado com tarefas de tempo de execução "mais curto".

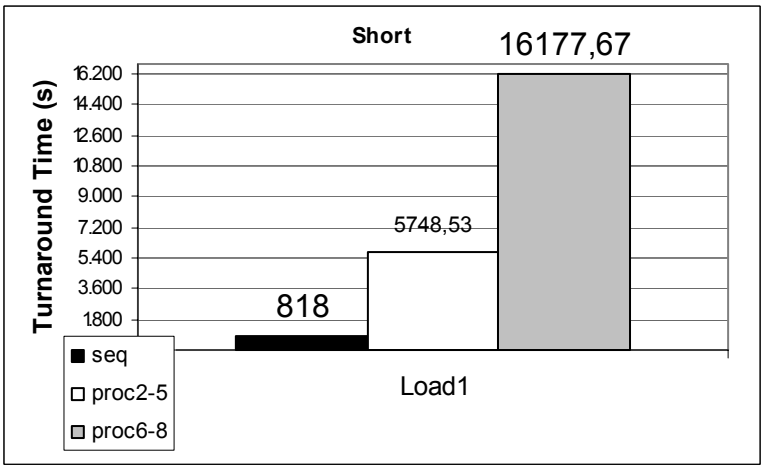

Figura 6. "Turnaround Time" médio para tarefas seqüenciais e paralelas com tempos de execução entre 1 e 10 segundos.

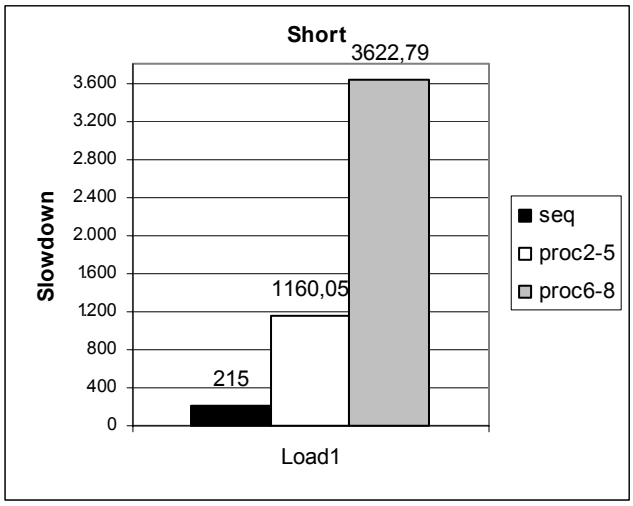

Figura 7. "Slowdown" para tarefas seqüenciais e paralelas com tempos de execução entre 1 e 10 segundos.

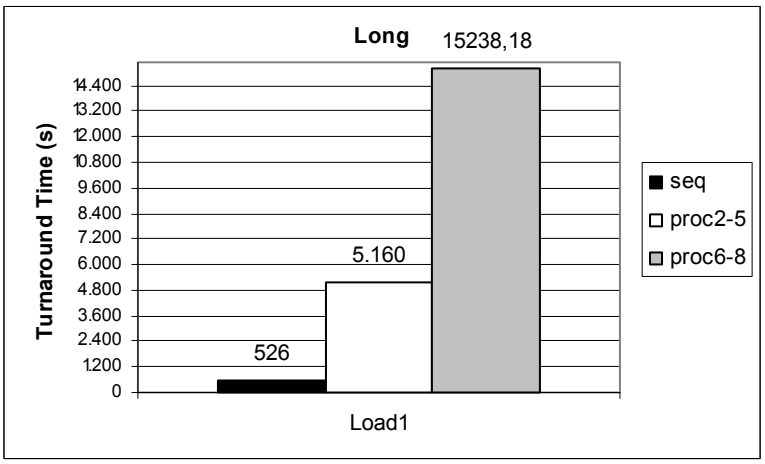

Figura 8. "Turnaround Time" médio para tarefas seqüenciais e paralelas com tempos de execução entre 10 e 60 segundos.

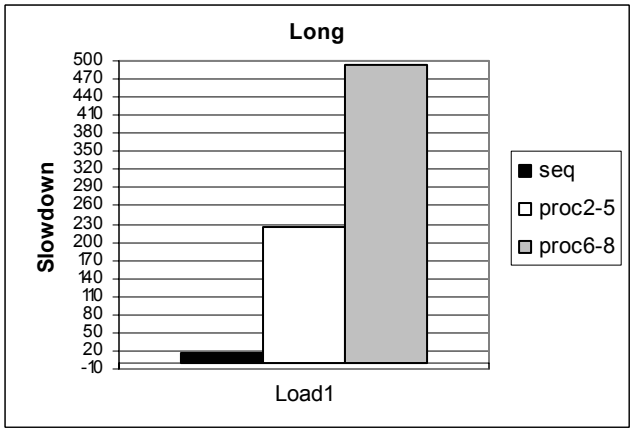

Figura 9. "Slowdown" para tarefas seqüenciais e paralelas com tempos de execução entre 10 e 60 segundos.

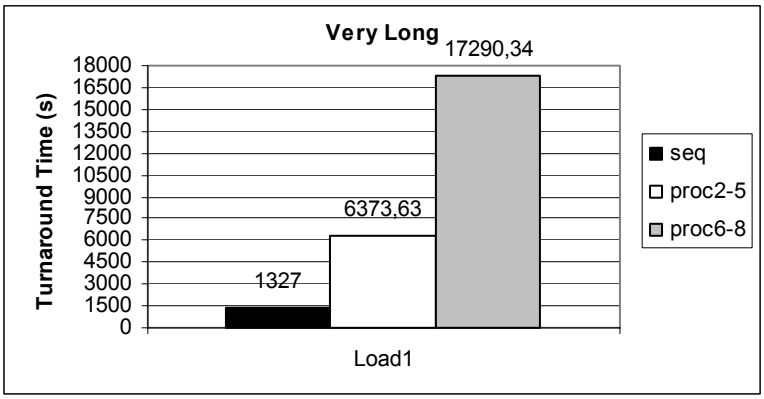

Figura 10. "Turnaround Time" médio para tarefas seqüenciais e paralelas com tempos de execução maiores que 60 segundos.

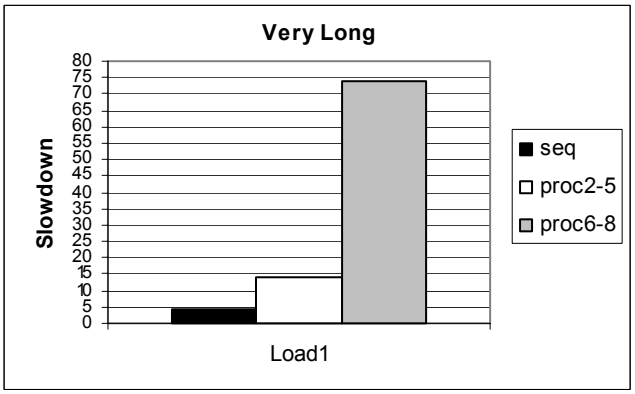

Figura 11. "Slowdown" para tarefas seqüenciais e paralelas com tempos de execução maiores que 60 segundos.

Para este tipo de carga, resultados mostram que tarefas longas apresentam menor tempo de espera ("slowdown" diminuiu) e que tarefas curtas apresentam uma sobrecarga de tempo muito alta, tanto em tempo de resposta quanto em tempo de espera. Também se pode concluir que o OpenPBS sempre proporcionou melhor desempenho para tarefas que demandam menor numero de processadores.

Em outras palavras, para este tipo de carga, o OpenPBS apresentou maior eficiência para tarefas que demandam menor número de processadores e com tempo de execução longo ou muito longo.

O algoritmo de escalonamento do OpenPBS é o FCFS (First Come First Serve), o qual escalona tarefas na medida que surgem, mas se a tarefa na fila requer um numero de processadores que não está disponível, o algoritmo suspende essas tarefas até que estejam disponíveis os processadores necessários.

4.1. Análise de Desempenho variando Carga

Com o objetivo de analisar o desempenho do OpenPBS em situações de carga diferente, foi reduzido o tempo de execução de todas as tarefas e foi mantida a mesma distribuição de carga com respeito ao número de processadores e aos tempos de chegada. A Figura 12 apresenta as duas distribuições utilizadas, a diferença entre elas é que o tempo de execução das tarefas para "carga2" ("load2") é maior do que para "carga3" ("load3"),

simulando desta maneira carga diferente para a maquina paralela. É importante salientar que tanto "carga2" quanto "carga3" apresentam o mesmo comportamento do que "carga1", ou seja, obedecem às distribuições das figuras 1 e 3.

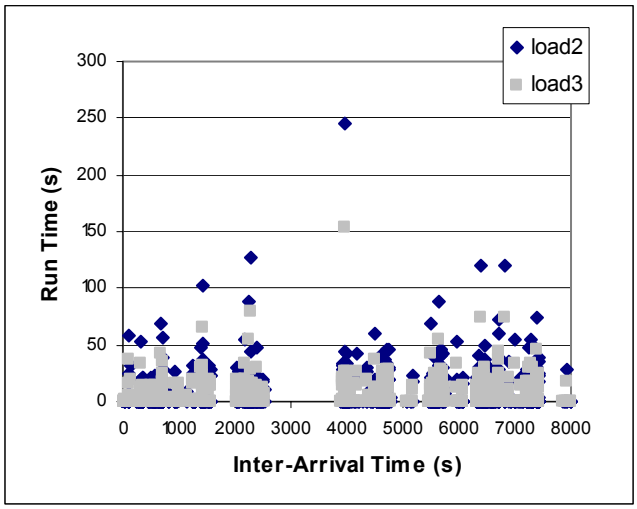

Figura 12. Duas distribuições de tarefas com tempos de execução diferentes

A Figura 13 mostra o "Turnaround Time" médio para tarefas seqüências, e paralelas para as duas cargas. Como se pode observar, o OpenPBS apresenta um melhor desempenho para "carga3" ilustrando que a redução do tempo de execução é refletido no melhor comportamento do OpenPBS de maneira geral. Neste caso também se mostra que as tarefas que demandam maior numero de processadores permanecem mais tempo na fila de espera demorando em ser executadas. A Figura 14 confirma esse fato.

As Figuras 15 e 16 mostram como o OpenPBS executou de maneira real as tarefas seqüenciais e as tarefas paralelas que usam 6 a 8 processadores para "carga2" e "carga3" respectivamente. Isto confirma as observações anteriores que o OpenPBS executou primeiro as tarefas seqüenciais. Pode-se também explicar o porque do "Slowdown" de "carga2" ser maior do que de "carga3" porque como ilustra a Figura 16 muitas mais tarefas de tamanho maior foram executadas antes do que no caso da Figura 15.

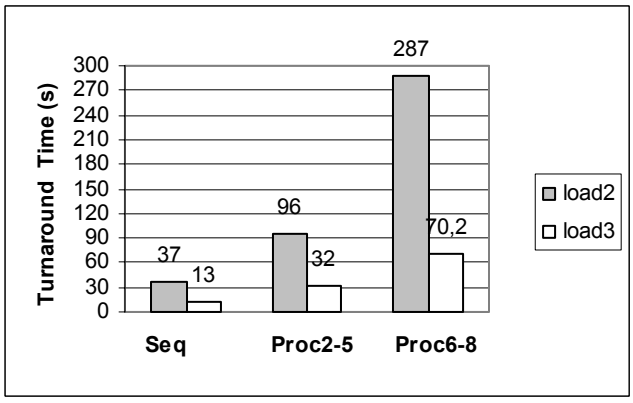

Figura 13. "Turnaround Time" médio para tarefas seqüenciais e paralelas

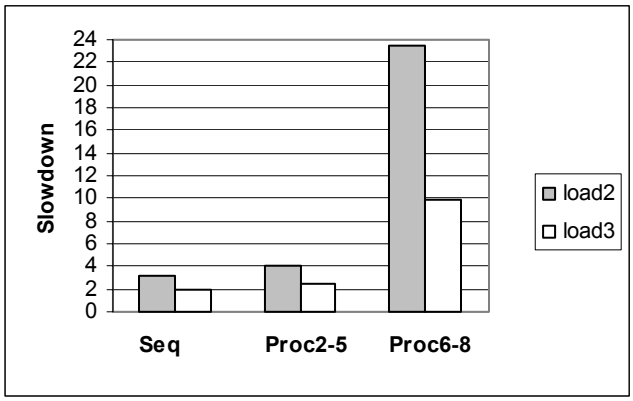

Figura 14. "Slowdown" para tarefas seqüenciais e paralelas

Os experimentos ilustrados neste item permitem concluir que na medida que se reduz a carga do sistema, o OpenPBS incrementa sua eficiência. Além disso, tarefas com menor número de processadores são priorizadas para sua execução.

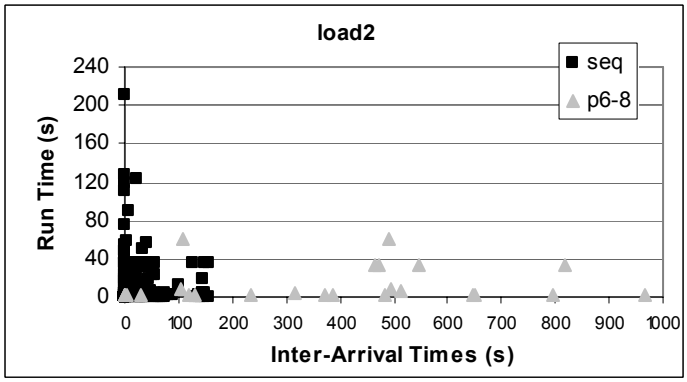

Figura 15. Distribuição real das tarefas seqüenciais e paralelas (de 6 até 8 processadores) feitas pelo OpenPBS quando submetido para a "carga2".

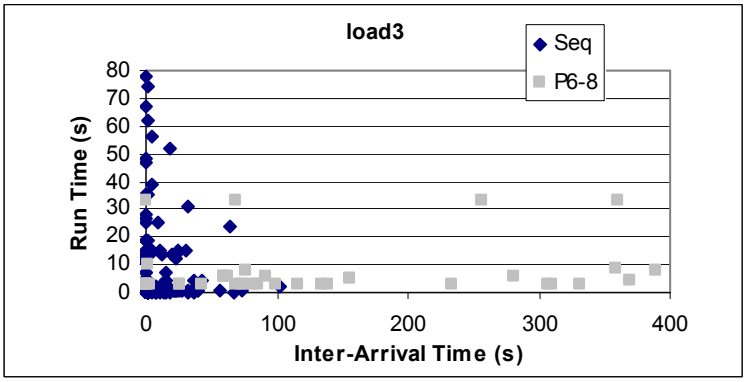

Figura 16. Distribuição real das tarefas seqüenciais e paralelas (de 6 até 8 processadores) feitas pelo OpenPBS quando submetido para a "carga3".

A Figura 17 mostra o "Slowdown" do sistema total para as diferentes "cargas". Isto em termos quantitativos quer dizer que o OpenPBS gera um tempo de espera muito grande quando submetido à "carga1" ou seja as tarefas permanecem muito tempo em

fila esperando para ser escalonadas. Usando “carga2” e “carga3” o tempo de espera é muito menor, já que o sistema está carregado de maneira mais leve.

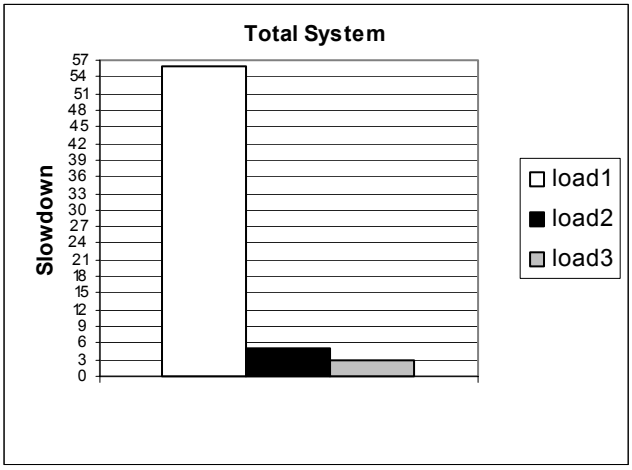

Figura 17. “Slowdown” total para cargas de trabalho diferentes

4.2. Analise de Desempenho do Overhead do OPenPBS

Com objetivo de medir quantitativamente o overhead que o OpenPBS apresenta, tem-se gerado sinteticamente várias cargas de trabalho que permitem proporcionar medidas importantes sobre o overhead do OpenPBS.

Em cada um dos casos, que serão apresentados, pretende-se analisar o desempenho do OpenPBS quando apresenta uma carga que ocupa todos os nós da máquina e que usa tarefas paralelas submetidas ao mesmo tempo e com tempos de execução previamente estabelecidos.

Inicialmente foram selecionadas três tarefas. Estas tarefas demandam número de processadores diferentes: Tarefa1 usa dois nós de processamento e seu tempo de execução é de 3,5 segundos, Tarefa2 usa três nós de processamento e seu tempo de execução é de 2,4 segundos, Tarefa3 usa quatro nós de processamento e seu tempo de execução é de 1,7 segundos.

Para a primeira analise, mostrada na Figura 18, escolhemos somente a Tarefa1, a qual primeiro é submetida individualmente, isto corresponde à carga de trabalho A, depois escolhemos duas destas mesmas tarefas e mandamos executá-las concomitantemente, correspondendo à carga de trabalho B. Continuamos aumentando a três tarefas (carga de trabalho C) e por último submetem-se quatro tarefas ao mesmo tempo (carga de trabalho D).

Na carga de trabalho A, o “Turnaround Time” foi de 5 segundos e o tempo de execução (Run Time) fornecido pelos nós de processamento foi de aproximadamente 3,5 segundos. Portanto pode-se dizer que neste caso específico o overhead do OpenPBS foi de 1,5 segundos.

Na carga de trabalho B, o “Turnaround Time” foi de 6,5 segundos, isto devido a que a segunda tarefa1 não foi executada ao mesmo tempo pelo OpenPBS, mesmo tendo os oito nós de processamento livres. A segunda tarefa1 esperou (Twait) 4 segundos para ser executada.

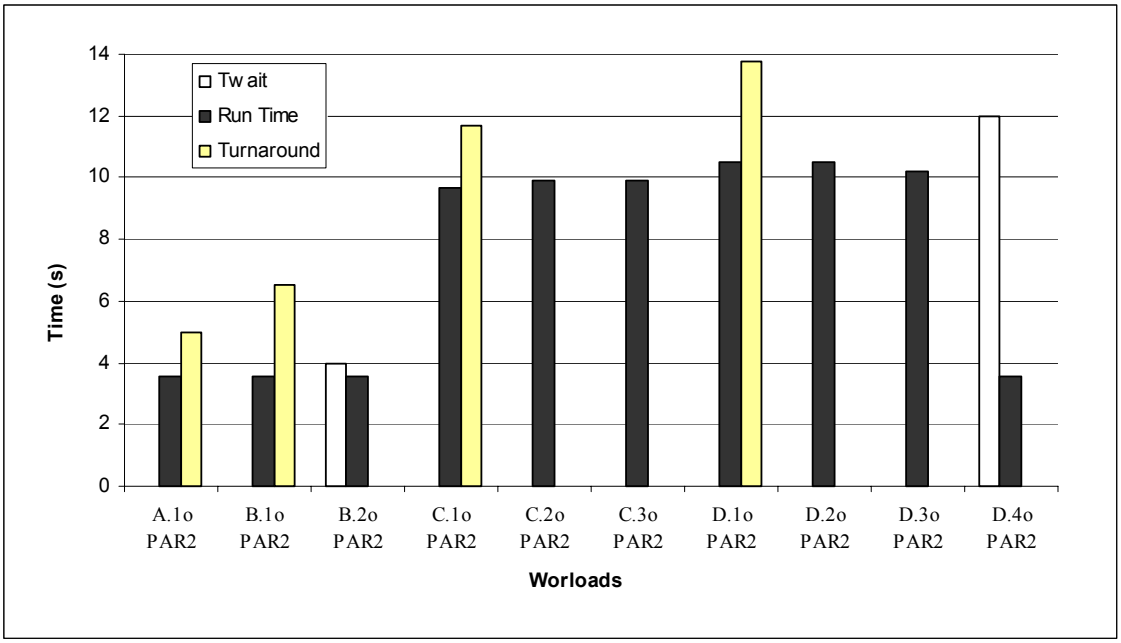

Figura 18. Overhead do OpenPBS para cargas de trabalho diferentes (A até D) com tarefas paralelas de dois processadores (PAR2) de tempo de execução constante.

Na carga de trabalho C, todos as tarefas foram executadas simultaneamente pelo OpenPBS já que o tempo de espera foi zero. A observação principal é que o tempo de execução (run time) das tarefas paralelas incrementou-se em aproximadamente 6,5 segundos. Isto também afetou a medida do "Turnaround Time". Neste caso observa-se que ao permitir a execução simultânea de três tarefas paralelas, o tempo de execução das tarefas é afetado.

Na carga de trabalho D, observa-se que a quarta tarefa não foi executada simultaneamente e apresenta um tempo de espera de 12 segundos. O tempo de execução do resto de tarefas, que foram executadas simultaneamente, foi incrementado.

Portanto da Figura 18 pode-se concluir que o OpenPBS ao executar simultaneamente varias tarefas paralelas de dois processadores adiciona um overhead considerável ao tempo de execução.

Para a segunda analise, mostrada na Figura 19, escolhemos a Tarefa1 e a Tarefa2. Na carga de trabalho A, duas tarefas do tipo Tarefa2 são submetidas simultaneamente, na carga de trabalho B, três tarefas são submetidas simultaneamente: duas tarefas tipo Tarefa2 e uma tarefa tipo Tarefa1.

Na carga de trabalho A, o tempo de execução das tarefas paralelas não foi afetado, mas observa-se que o OpenPBS adicionou um tempo de espera para submeter a segunda tarefa, ou seja as duas tarefas não foram executadas simultaneamente. Portanto o "Turnaround Time" também é afetado.

Na carga de trabalho B, as tarefas foram executadas simultaneamente e o tempo de execução foi incrementado.

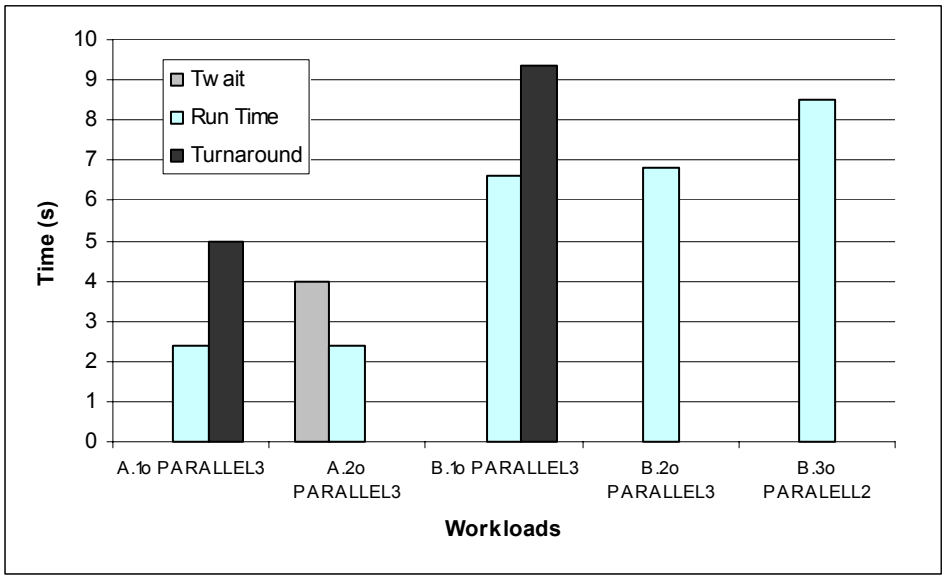

Figura 19. Overhead do OpenPBS para cargas de trabalho diferentes (A e B) com tarefas paralelas de dois e três processadores (PARALELO2 e PARALELO3) de tempo de execução constante.

Da Figura 19 pode-se concluir novamente que o OpenPBS influencia o desempenho das tarefas paralelas quando estas se executam ao mesmo tempo.

Para a terceira analise, mostrada na Figura 20, escolhe-se a Tarefa3. Na carga de trabalho A, uma tarefa do tipo Tarefa3 é submetida individualmente; na carga de trabalho B, duas tarefas tipo Tarefa3 são submetidas simultaneamente.

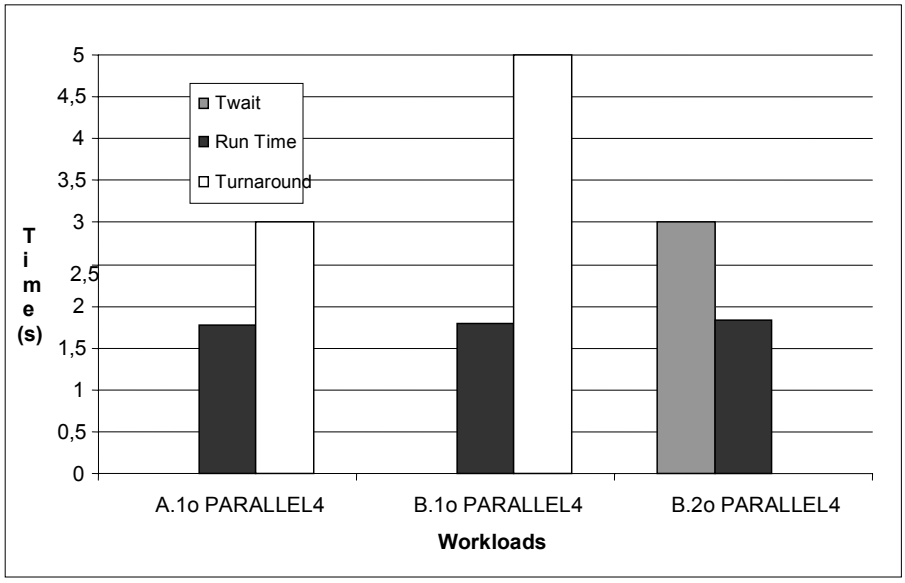

Figura 20. Overhead do OpenPBS para cargas de trabalho diferentes (A e B) com tarefas paralelas de quatro processadores (PARALELO4) de tempo de execução constante.

Estas cargas de trabalho apresentam comportamento similar do caso anterior. Na carga de trabalho A, o tempo de execução da tarefa não é afetado e o OpenPBS apresenta um overhead de aproximadamente 1,5 segundos. Na carga de trabalho B o

tempo de execução também não é afetado, mas existe um tempo de espera para executar a segunda tarefa que faz com que o "Turnaround Time" se incremente.

É importante salientar que estes experimentos também foram executados sem o uso do OpenPBS e em todos os casos o tempo de execução das tarefas paralelas não foi afetado conservando o valor esperado. Ou seja, a máquina paralela conseguiu executar simultaneamente as tarefas paralelas sem incrementar o tempo de execução. Fato que não ocorre quando o OpenPBS realiza o escalonamento das tarefas.

Dos resultados obtidos pode-se concluir que o OpenPBS incrementa o tempo de resposta e o tempo de espera das tarefas quando gerencia ao mesmo tempo tarefas paralelas no sistema.

4.3. Analise de Desempenho do comportamento do escalonador do OPenPBS

Para ilustrar a política de escalonamento no OpenPBS criamos a carga de trabalho mostrada na Figura 21, que consiste de sete tarefas: P1 (seqüencial, tempo de execução: 8.5 seg), depois de dois segundos é submetida a tarefa P8 (oito processadores, tempo de execução: 0.9 seg), depois de um segundo é submetida a tarefa P7 (sete processadores, tempo de execução: 1 seg), depois de um segundo é submetida a tarefa P2 (dois processadores, tempo de execução: 6.2 seg), depois de um segundo é submetida a tarefa P3 (três processadores, tempo de execução: 4,9 seg), depois de um segundo é submetida a tarefa P1 (seqüencial, tempo de execução: 7 seg) e depois de um segundo é submetida a tarefa P4 (quatro processadores, tempo de execução: 1.7 seg).

A Figura 22 mostra a ordem de execução que o OpenPBS executou para este tipo especial de carga de trabalho.

Deste resultado podemos observar que embora a segunda tarefa na fila seja a tarefa P8 esta não foi executada imediatamente depois de executar a primeira tarefa P1. O OpenPBS escalonou a tarefa P7 assim que ela foi submetida já que tinha 7 nós de processamento livres. Depois executa as tarefas P2, P3 e P1 e assim que a máquina teve quatro nós livres, escalonou a tarefa P4 e por último executou a tarefa P8.

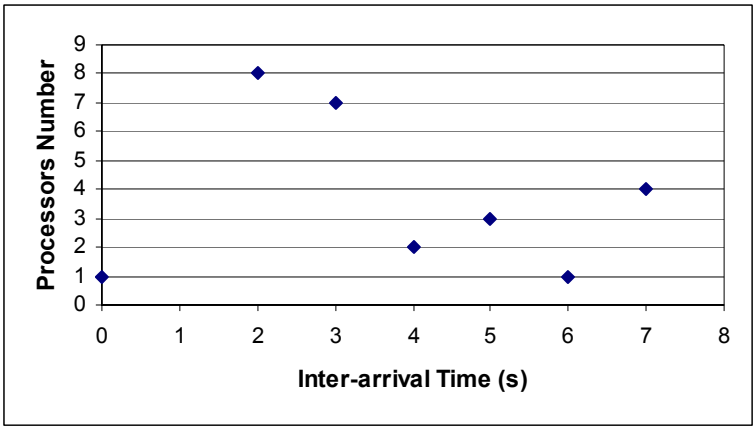

Figura 21. Distribuição das tarefas ao longo do tempo

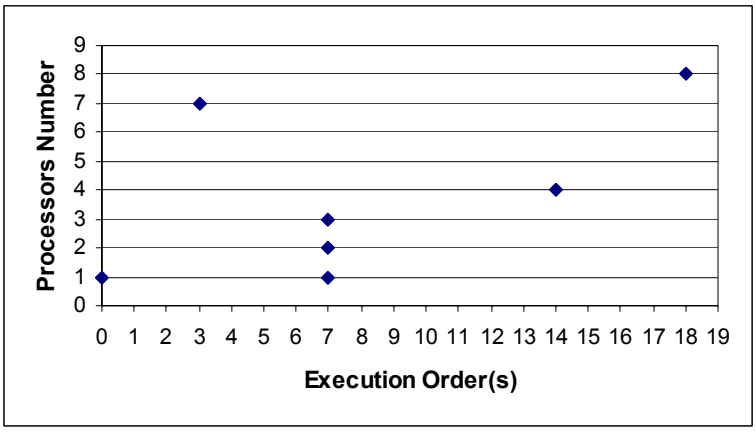

Figura 22. Ordem de execução realizada pelo OpenPBS

Este exemplo ilustra como o escalonador do OpenPBS respeitou a política de escalonamento FCFS (First Come First Serve), escalonando as tarefas na sua ordem de chegada mas sempre respeitando a capacidade de nós da máquina. Este experimento também ilustra um caso específico onde a tarefa que demandou maior número de processadores foi executada por último.

5. Conclusões e Trabalhos Futuros

Neste trabalho foi realizada a avaliação de desempenho quantitativa do sistema de gerenciamento de tarefas OpenPBS sobre uma máquina paralela baseada em aglomerados de PCs, usando cargas de trabalho dinâmicas e estáticas.

Para cargas de trabalho dinâmicas, foi avaliado o desempenho em função do tamanho das tarefas, com respeito ao tempo de execução e com respeito ao número de processadores. Também foi avaliado o desempenho em função da carga do sistema.

Desta análise pode-se concluir que o OpenPBS, quando baseado no algoritmo de escalonamento FCFS, apresenta melhor desempenho para tarefas que demandam menor número de processadores. Tarefas longas apresentam menor tempo de espera e sistemas carregados mais levemente são mais eficientes.

As medidas de desempenho efetuadas com cargas de trabalho estáticas permitiram ilustrar que o OpenPBS diminui seu desempenho quando executa tarefas paralelas ao mesmo tempo. Estes experimentos também mostraram que o OpenPBS apresenta sempre um overhead de poucos segundos.

Foi desenvolvido um experimento que permitiu ilustrar o comportamento do escalonador FCFS do OpenPBS.

Tanto a metodologia usada quanto as ferramentas desenvolvidas neste trabalho podem ser utilizadas para avaliar o desempenho do OpenPBS em outras plataformas paralelas, avaliar outros algoritmos de escalonamento que possam ser implementados no OpenPBS e ser usados como base para avaliar outros sistemas de escalonamento de tarefas.

Como trabalhos futuros espera-se realizar uma análise comparativa com outros algoritmos de escalonamento implementados no OpenPBS e também realizar comparações de desempenho com outros aglomerados de PCs.

6. Agradecimentos

Os autores agradecem à CAPES e ao CNPq pelo apoio financeiro e ao laboratório de bioinformatica de DCC-IME-USP pela infra-estrutura.

7. References

Kento Aida, “Effect of Job Size Characteristics on Job Scheduling Performance”. In: Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science Vol. 1911, Springer-Verlag, pp.1-17, 2000.

Carsten Ernemann, Volker Hamscher, Uwe Schwiegelshohn, Ramin Yahyapour, Achim Streit, “On Advantages of Grid Computing for Parallel Job Scheduling”. In d

Eitan Frachtenberg, Dror G. Feitelson, Juan Fernandez, and Fabrizio Petrini, “Parallel Job Scheduling Under Dynamic workloads”. In *Job Scheduling Strategies for Parallel Processing*, Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, (ed.), Springer Verlag, Lect. Notes Comput. Sci. vol. 2862, pp. 208--227, 2003.

Eitan Frachtenberg, Fabrizio Petrini, Juan Fernandez, Scott Pakin, and Salvador Coll, “STORM: Lightning-Fast Resource Management”. In *Supercomputing 2002.*

Kris Gaj, Nguyen Nguyen, Jacek R. Radzikowski, Tarek El-Ghazawi, Nikitas Alexandridis, Frederic Vroman, Preeyapong Samipagdi, Suboh A. Suboh, “Performance Evaluation of Selected Job Management Systems”. In International Parallel and Distributed Processing Symposium: IPDPS 2002 Workshops

Rajkumar Kettimuthu, Vijay Subramani, Srividya Srinivasan, Thiagaraja Gopalasamy, “Selective Preemption Strategies for Parallel Job Scheduling”. In 2002 International Conference on Parallel Processing (ICPP'02)

Uri Lublin, Dror G. Feitelson, “The workload on parallel supercomputers: modeling the characteristics of rigid jobs”. In: Journal of Parallel and Distributed Computing,1105-1122 Volume 63, Number 1, January 2003

Jahanzeb Sherwani, Nosheen Ali, Nausheen Lotia, Zahra Hayat and Rajkumar Buyya, “Libra: An Economy driven Job Scheduling System for Aglomeradoss”. In HPC Asia 2002

Martha Torres, Alfredo Goldman, Junior Barrera, “A Parallel Algorithm for Enumerating Combinations”. In: 2003 International Conference on Parallel Processing.