6
Resultados

Neste capítulo apresenta-se uma descrição completa do problema do cálculo da trajetória de partícula assim como um detalhamento dos operadores desenvolvidos para suportar o processamento desta aplicação. Ao final descrevem-se alguns dos resultados obtidos.

6.1. Traçado de Partículas

A aplicação de visualização científica utilizada neste trabalho é o cálculo da trajetória de partículas (CTP) que determina a posição de um conjunto de partículas em cada instante de tempo ou iteração. O desafio é minimizar o tempo decorrido no cálculo das trajetórias, o qual pode ser elevado dependendo da quantidade de partículas e iterações que cada partícula deve realizar. Este problema pode ser descrito matematicamente por (L.ROSENBLUM & ET AL., 1994):

$$\begin{aligned} &\frac{dx}{dt} = \vec{F}(x,t), \\ &x(0) = P0 \end{aligned} \qquad (8)$$

onde $\vec{F} : \Re^3 \times \Re^+ \to \Re$, é um vetor dependente do tempo (i.e. velocidade). A solução do problema (1) resulta em uma curva, que pode ser interpretada como a trajetória de uma partícula sem massa definida pelo fluxo gerado pelo campo $\vec{F}$.

A representação de banco de dados utilizada na aplicação de cálculo de trajetória de partícula pode ser descrita por um modelo objeto-relacional composto das relações: Partícula, Geometria e Velocidade.

Nas aplicações de interesse, modelos matemáticos baseados em equações de fluidos (equações de Navier-Stokes (GILSON GIRALDI, ANTÔNIO LOPES APOLINÁRIO JR et al., 2005)) são resolvidos numericamente, fornecendo como resultado uma versão discreta do campo de velocidades do

fluido. Para isto, os modelos matemáticos em questão (contínuos), precisam ser discretizados para efetuar as simulações numéricas. O resultado desta discretização é a substituição do meio contínuo inicial do modelo por um conjunto discreto e finito de pontos no espaço e no tempo nos quais as derivadas que aparecem nas equações serão aproximadas. Nos métodos tradicionais de Elementos Finitos e Diferenças Finitas, este conjunto de pontos é conectado seguindo uma topologia conveniente o que dará origem a uma malha que pode ser de um dos três tipos:

- Malha Estruturada: cada ponto tem o mesmo número de pontos vizinhos.
- Malha Não-Estruturada: O número de pontos vizinhos é variável.
- Malha Mista: Parte da malha é estruturada e parte é não-estruturada.

Nos exemplos a seguir, é usada uma malha não-estruturada (geometria), formada por tetraedros. A simulação numérica das equações de fluidos fornece o vetor velocidade em cada nó da malha utilizada, para cada instante de simulação.

Esta versão discreta do campo de velocidades é usada para movimentar as partículas, segundo a solução numérica da equação (8), segundo o fluxo definido pelo campo (discreto) de velocidades. Esta solução é obtida via um método iterativo que usa a solução de um instante como condição inicial para o cálculo no instante seguinte. Se usarmos um conjunto de partículas, então teremos um conjunto de condições iniciais, cada uma das quais gerando uma trajetória independente.

6.1.1. Esquema de Banco de Dados

A relação Geometria define dados a cerca dos tetraedros que formam a geometria (malha). A relação Velocidade, por sua vez, contém informações sobre o vetor velocidade de cada vértice do tetraedro em um dado instante de tempo. E, por fim, a relação Partícula define a posição de um conjunto de partículas. O modelo de banco de dados para esta aplicação é apresentado em detalhes a seguir:

- Ponto (x:float, y:float, z:float)

- Partícula (id:int, posição:Ponto, tempo:int)
- Geometria (idTetraedro:int, tetraedro:Lista<id:int, posição:Ponto>)
- Velocidade (idTetraedro:int, tempo:int, vetorVelocidade:Lista <vetor:Ponto>)

Neste modelo, cada um dos vértices de um tetraedro e as posições de cada partícula são representados por uma terna de números reais (x,y,z) e, um identificador é associado a cada objeto para permitir a correlação entre os mesmos durante a execução.

O problema descrito anteriormente consiste em determinar a próxima posição de uma partícula repetidas vezes e pode ser dividido em três etapas: (1) encontrar o tetraedro no qual a partícula está inserida em um dado instante de tempo, (2) encontrar o vetor velocidade de cada ponto do tetraedro também neste instante de tempo e, por fim, (3) determinar o vetor velocidade resultante a partir dos quatro vetores encontrados anteriormente e calcular a próxima posição da partícula. Do ponto de vista de processamento de consulta, as operações (1) e (2) podem ser implementadas a partir da extensão das operações de junção convencionais, com funções espaciais, ao passo que a operação (3) pode ser definida como um programa do usuário que recebe como parâmetro a posição atual da partícula e os vetores velocidade de cada ponto do tetraedro.

O cálculo associado a cada iteração pode ser expresso por uma consulta SQL como a apresentada na Figura 13. A noção de múltiplas iterações requer a execução desta consulta internamente a um laço em uma linguagem SQL estendida.

```
Select P.id, TCP(P.Posição, V.vetorVelocidade)
From Geometria G, Velocidade V,   Particula P
Where G.inside(P.posição) and
        G.idTetraedro = V.idTetraedro and
        V.tempo = $tempo
```

Figura 13. Consulta SQL para o CTP

6.1.2. Suporte ao CTP

Para suportar a aplicação de visualização científica mencionada anteriormente algumas extensões foram realizadas na máquina de execução

proposta inicialmente, são elas: suporte a execução de programas do usuário, desenvolvimento de algoritmos de junções espaciais e temporais baseados em *hash*.

A estratégia utilizada no suporte a execução de programas do usuário foi a implementação do operador Apply (FABIO PORTO, GILSON GIRALDI et al., 2004) como um novo operador algébrico que implementa a interface padrão de um iterador (G.GRAEFE, 1993). Este operador encapsula a invocação de programas do usuário e sua entrada é obtida através do mapeamento dos valores de uma tupla.

A necessidade de algoritmos de junção baseados em *hash* surge do fato que a geometria e os dados temporais associados a ela podem ser grandes e, consequentemente, pode não haver memória principal suficiente para que eles sejam carregados. Uma boa função de *hash* particionaria os dados em *buckets* de maneira a reduzir o número de I/Os. Por exemplo, a função de particionamento do vetor velocidade poderia considerar que as partículas evoluem em conjunto e, sendo assim, agruparia em um *bucket* dados de iterações consecutivas.

A implementação do algoritmo de *hash* utilizado segue a idéia de um *framework* proposto em (LO M-L & RAVISHANKAR CV., 1996), onde diferentes funções de hash poderiam ser utilizadas. Em função da característica circular do problema, a utilização de algoritmos de junção que realizam o particionamento de ambas as relações durante a fase de inicialização é inviável. Na junção espacial entre uma partícula e seu tetraedro, por exemplo, o algoritmo ficaria aguardando eternamente o final da relação de partículas. Para resolver este problema, o algoritmo implementado realiza apenas o particionamento da relação *inner* durante a inicialização, o particionamento da relação *outer* é realizada durante a execução por uma linha de execução independente.

A estratégia de paralelização adotada considera que todas as instâncias do operador de junção receberão todos os dados sobre a geometria e vetores velocidade, desta forma, uma partícula independe da instância do operador e pode ser escalonada de acordo com o desempenho de cada nó.

6.2. Resultados Obtidos

Nesta seção apresentam-se alguns dos resultados obtidos a partir de uma aplicação que simula o cálculo da trajetória de partícula. Apesar da implementação realizada oferecer suporte ao CTP, esta aplicação não foi utilizada para facilitar a análise dos resultados.

Os testes realizados ilustram os principais problemas relacionados à execução de consultas iterativas no ambiente de grade e, destacam a capacidade de reação do modelo de adaptação proposto no capítulo anterior.

Para simular a aplicação de visualização científica utilizou-se a infra-estrutura descrita no capítulo 4 com apenas um operador *dummy* nos nós de execução. Este é um operador de controle que permite configurar a taxa de processamento de cada nó (ms/tupla) em trechos da execução.

O módulo de execução de consulta desenvolvido para o CODIMS-G utiliza a linguagem de programação Java 1.4.2 e o *container* de *serviços web* que acompanha o Globus 3.2.1. Os testes aqui descritos foram realizados no *cluster* Carcará (Carcara) que é suportado pelo Laboratório Nacional de Computação Científica. No total foram utilizadas dezesseis máquinas com processadores Pentium IV 2.8GHz , 512 MB de memória e sistema operacional Linux Slackware 2.6.8.1.

6.3. Speedup

O experimento a seguir tem por objetivo determinar a escalabilidade da solução e verificar a proximidade entre o tempo total de execução previsto pelo G2N e o tempo real de execução. O experimento realizado utiliza um conjunto de um, seis, onze e dezoito nós para processar mil partículas durante vinte e cinco iterações. Cada nó processa os dados a uma taxa de 220 ms/tupla e, de acordo com a distribuição realizada pelo G2N, cada um deverá processar cem partículas.

O resultado apresentado pela figura 14 mostra que o tempo de execução real é bem próximo ao previsto pelo G2N e que a solução começa a sofrer problemas de escalabilidade com aproximadamente dezoito nós.

A escalabilidade da solução é limitada pela política de distribuição dos dados utilizada no nó central que, realiza a construção dos blocos de forma serial para evitar a ocorrência de *timeouts* durante a montagem de vários blocos quando

o volume de dados é reduzido. Uma análise mais detalhada, mostra que a escalabilidade da solução está relacionada à fração entre B_i Nt_i e o tempo gasto na montagem dos blocos, quanto maior for o resultado desta fração, maior pode ser o número de nós alocados em uma execução.

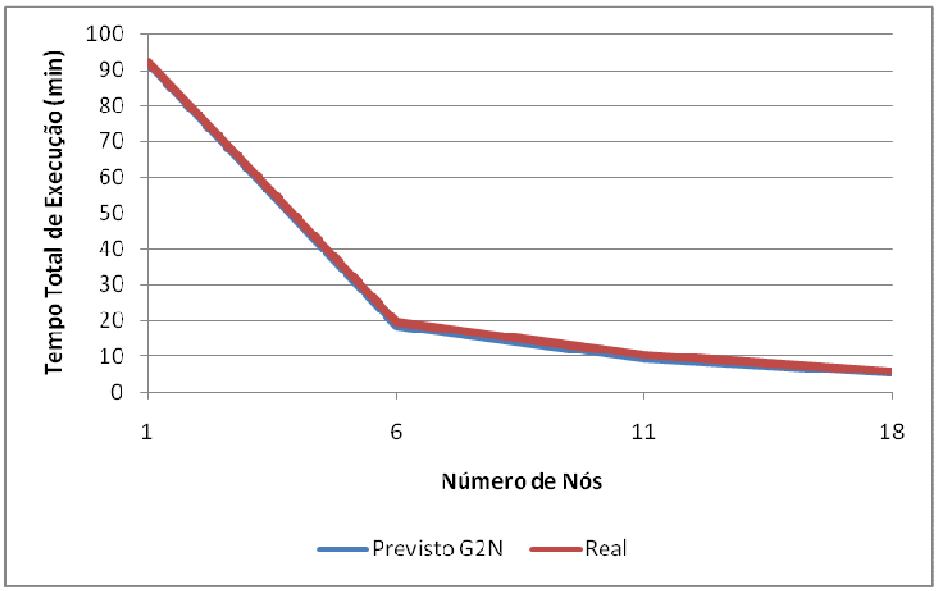

Figura 14. Escalabilidade da Execução

6.3.1. Uma Primeira Comparação

Os testes desta seção são realizados com um conjunto de onze nós (dez nós de execução), mil partículas e vinte iterações. Inicialmente, pressupõe-se que todos os nós possuem uma mesma taxa de execução e é realizada uma distribuição homogênea dos dados. Cada nó de execução deverá processar um conjunto de cem partículas e utilizará blocos de tamanho $Q_i/2$. Após o processamento de um conjunto de iterações as taxas de processamento de cada nó são modificadas, simulando um ambiente no qual a execução pode estar compartilhando recursos com outras tarefas.

O primeiro experimento é realizado em quatro cenários diferentes, em cada um deles varia-se o número de nós que apresentam alterações de desempenho. Até a metade da execução todas as taxas se mantêm como previstas inicialmente, a partir daí, um conjunto de zero, três, seis e nove nós têm suas taxas alteradas de 500 ms/tupla para 1500 ms/tupla. Para cada cenário executam-se os testes

utilizando três níveis de adaptação diferentes: sem adaptação, adaptação ao desempenho e adaptação ao desempenho e volume de dados. A figura 15 apresenta os resultados deste experimento.

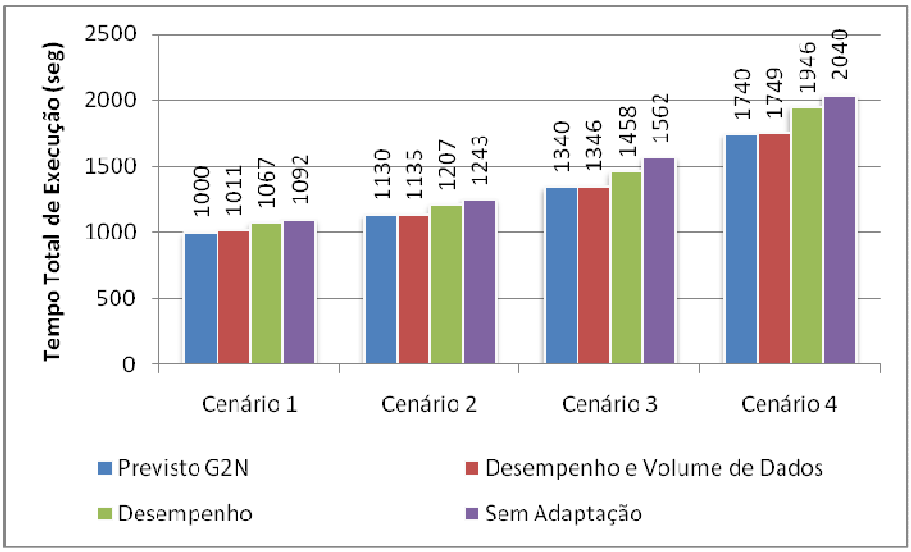

Figura 15. Alterações de Desempenho x Nível de Adaptatividade

A partir do gráfico apresentado na figura 15 pode-se concluir que a adaptação apenas à variação nas taxas de processamento cada nó não possui resultados expressivos. Entretando, quando combinada a adaptação por volume de dados, resultados bem próximos do ideal podem ser alcançados. O ganho varia de 8% no cenário um a 17% no cenário quatro quando comparado a uma execução não adaptativa.

Nas próximas seções são realizados experimentos mais detalhados que isolam o problema associado a cada execução.

6.3.2. Adaptação às Variações de Desempenho

O próximo experimento apresenta uma visão mais detalhada da execução e tem o objetivo de validar a equação (5). Neste experimento utiliza-se o mesmo conjunto de nós e volume de dados, mas reduz-se para quinze o número de iterações realizadas.

Dois cenários de execução são avaliados: no primeiro, nove nós de execução possuem sua taxa de processamento alteradas de 300 ms/tuplas para 640 ms/tupla em um intervalo que vai da quinta a décima primeira iteração. No segundo

cenário, cinco nós possuem sua taxa de processamento alteradas de 300 ms/tupla para 500 ms/tupla enquanto a taxa dos outros nós é alterada de 300 ms/tupla para 100 ms/tupla durante o mesmo intervalo do cenário anterior. Durante uma iteração, cada nó processará um volume não superior a 100 partículas, após este número a taxa da iteração seguinte é utilizada.

A política de criação dos blocos é a mesma do primeiro experimento e, considerando que as taxas dos nós são iguais no instante inicial, cada nó receberá um volume inicial de cem partículas.

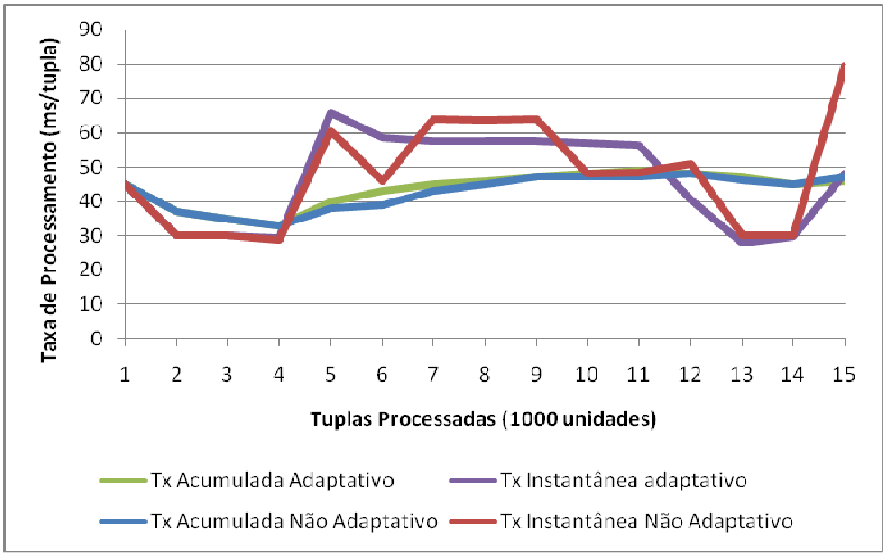

Figura 16. Desempenho do sistema em cenário com aumento da taxa de processamento

A figura 16 ilustra as taxas de processamento instantânea e acumulada do sistema a cada mil tuplas processadas para uma execução no cenário um. A taxa instantânea das duas execuções permanece igual até a 4ª iteração. Na execução adaptativa, a taxa de processamento se estabiliza em aproximadamente duas iterações sempre que há uma alteração de desempenho. Este intervalo pode ser maior ou menor de acordo com o tamanho da janela utilizada na captura das taxas de cada nó.

Na execução não adaptativa, a taxa de processamento pode variar durante todo intervalo em que as taxas de cada nó são diferentes. Neste caso, o processamento das iterações em um nó não é sincronizado com os demais e, sendo

assim, durante um intervalo (1000 tuplas) um nó pode contribuir com mais tuplas que os demais, distorcendo a taxa de cada instante.

Enquanto o volume de dados do sistema permanece constante, a falta de sincronia entre o processamento de cada iteração não impacta o tempo total de execução. Entretanto, este fato faz com que algumas partículas completem seu número de iterações primeiro que as outras, reduzindo o volume de dados do sistema de forma antecipada quando comparado à execução adaptativa. Como nesta execução não existe adaptação ao volume de dados, o problema da disputa por dados se agrava e deteriora o tempo de execução.

Na execução adaptativa a disputa por dados também prejudica o tempo total de execução. Em ambos os casos, quando este problema ocorre a execução passa a ocorrer de forma seqüencial, pois o tamanho dos blocos não é atualizado. Este problema pode ser comprovado pelo elevado número de *timeouts* que ocorre ao final da execução.

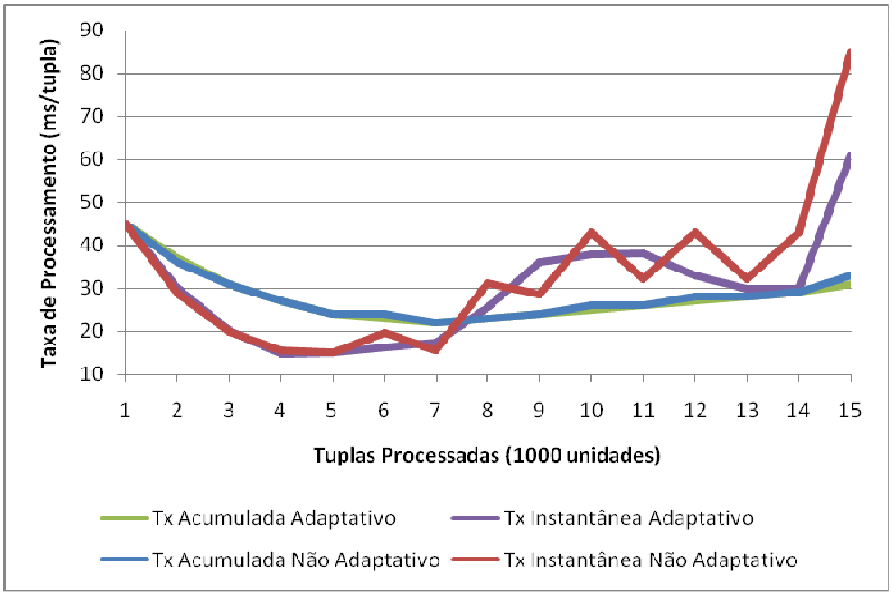

Figura 17. Desempenho do sistema em cenário com aumento e queda das taxas de processamento

A figura 17 ilustra a execução de uma consulta no cenário dois. Durante o segundo intervalo as taxas de processamento dos nós são diferentes e, em uma execução ótima, os cinco nós mais rápidos recebem um volume de dados superior. Considerando que as taxas de execução se alteram após cem tuplas processadas

podemos dividir o segundo intervalo em dois momentos: no primeiro, os cinco nós mais rápidos processam um volume de setecentas partículas a uma taxa de 100 ms/tupla. No segundo, os cinco nós mais rápidos teriam sua taxa normal restabelecida enquanto os outros nós ainda processam os dados a uma taxa de 500 ms/tupla. No gráfico ilustrado pela figura 17 estes dois momentos são representados pelo intervalo [4,8[e]8,11] da taxa de processamento instantâneo da execução adaptativa. A execução não adaptativa possui um comportamento semelhante, exceto pelas variações causadas pela falta de sincronia no processamento das iterações entre os nós.

Em ambos os cenários de execução podem-se perceber que o sistema foi capaz de adaptar a execução as variações das taxas de desempenho em aproximadamente uma iteração. Ao final de todas as execuções o sistema apresentou um aumento significativo em função da saída dos dados. Este problema á maior nas execuções não adaptativas, pois algumas partículas completam seu processamento antes que as outras e uma saída prematura dos dados é observada.

6.3.3. Adaptação às Variações no Volume de Dados

O próximo experimento tem como objetivo medir a capacidade de adaptação do sistema frente a variações no volume de dados disponível. Este experimento utiliza um conjunto de dez máquinas e mil partículas, o número de iterações processadas varia de acordo com os dois cenários avaliados. A taxa de processamento é igual para todos os nós (250 ms/tupla) e cada máquina deverá processar um conjunto de 100 partículas por iteração. A política de criação dos blocos é mantida em $Q_i/2$.

No primeiro cenário realiza-se uma diminuição acentuada no volume de dados disponível logo no início da execução. Em seguida são realizadas reduções menores de duzentas e cem partículas a cada quatro iterações. A figura 18 ilustra a saída de dados neste cenário.

A figura 19 ilustra o desempenho da execução neste cenário. A taxa de processamento instantâneo da execução adaptativa apresenta algumas alterações nos pontos próximos às reduções do volume de dados. Isso ocorre em função da

alteração no tamanho do bloco mas, apesar disso, pode-se observar que a sua taxa de processamento acumulada se mantém constante.

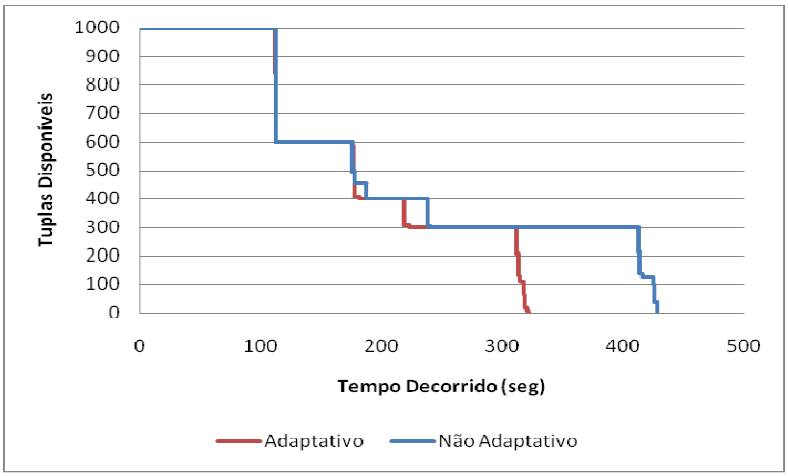

Figura 18. Tuplas disponíveis no sistema em cenário com redução acentuada do volume de dados no início da execução

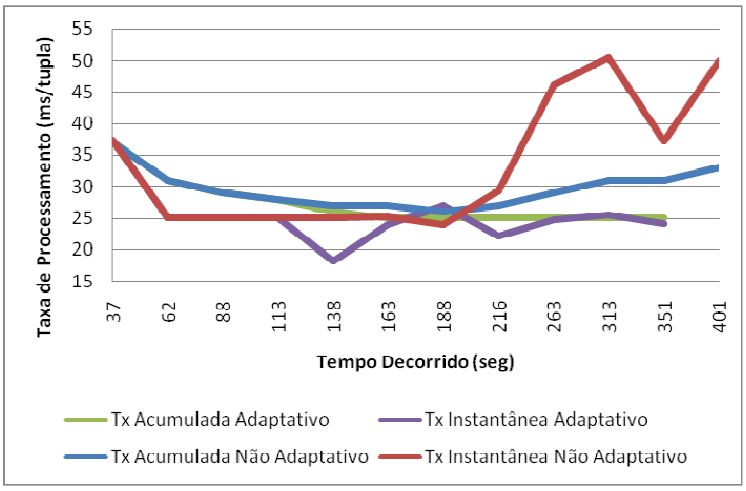

Figura 19. Desempenho do sistema em cenário com redução acentuada do volume de dados no início da execução

Já a taxa de processamento instantâneo da execução não adaptativa apresenta um aumento expressivo quando o volume de dados é reduzido pela metade. Neste ponto não é possível atender as solicitações por dados dos fragmentos de execução, pois a soma do tamanho dos blocos utilizados é maior que o volume de dados disponível. Em um cenário onde o custo de comunicação

é expressivo, este aumento pode acontecer ainda antes, pois o fragmento teria que esperar que os dados processados por ele fossem enviados ao nó principal.

No segundo cenário, a redução de dados inicial é pequena e aumenta durante a execução. Ela é realizada de quatro em quatro iterações e o número de tuplas removidas é igual a cem, duzentos e quatrocentos respectivamente. A figura 20 ilustra a saída de tuplas no sistema durante a execução.

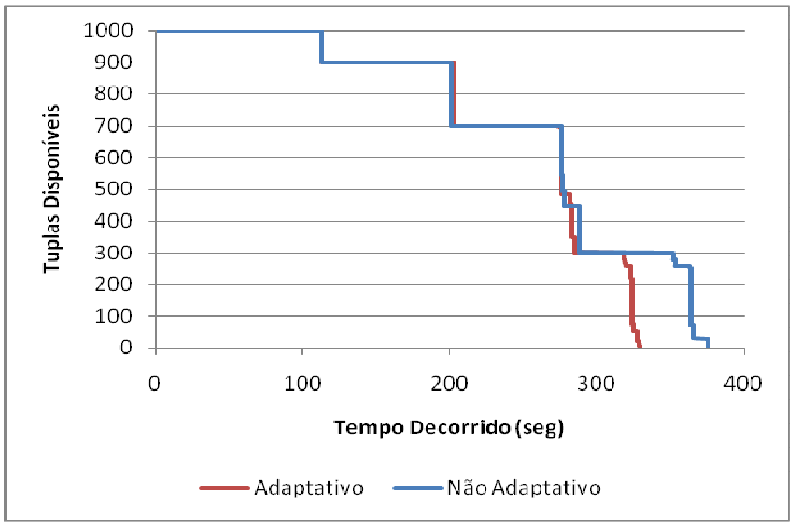

Figura 20. Tuplas disponíveis no sistema em cenário com redução acentuada do volume de dados no final da execução.

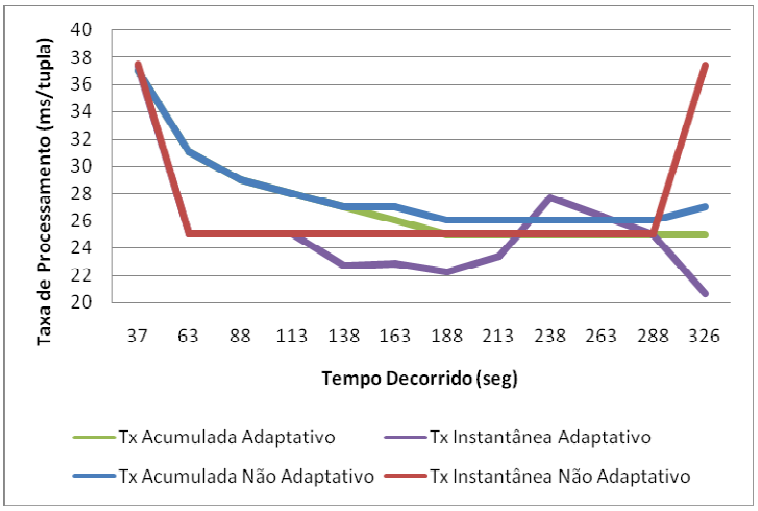

Figura 21. Desempenho do sistema em cenário com redução acentuada do volume de dados no final da execução.

A figura 21 ilustra a variação das taxas de desempenho neste cenário, por ela também podemos observar que a execução não adaptativa só apresenta problemas quando o volume de dados é reduzido pela metade. A diferença aqui é que este ponto ocorre no final de execução e, sendo assim, a taxa acumulada é relativamente menor.

6.3.4. Síntese do Capítulo

De acordo com os resultados dos experimentos apresentados neste capítulo pode-se observar que apenas a técnica de adaptação ao desempenho dos nós não é suficiente para garantir um bom tempo total de execução. Mas, quando combinada com a técnica de adaptação ao volume de dados verifica-se um grande potencial de adaptação, permitindo tempos de execução bem próximos ao determinado pelo G2N.

A utilização da técnica de adaptação ao desempenho permite que a execução das iterações possam ser executadas de forma síncrona entre os nós que participam da execução, evitando a saída de dados prematura do sistema. Em alguns cenários ela pode evitar que o grau de paralelismo seja reduzido.

A adaptação frente ao volume de dados disponível evita o problema conhecido na literatura como disputa por dados. Neste trabalho, este tipo de adaptação é extremamente importante, pois permite a atualização do tamanho dos blocos utilizados, evitando a ocorrência de *timeouts* e que a execução ocorra de forma serial.