CAP-387(2016) – Tópicos Especiais em Computação Aplicada: Construção de Aplicações Massivamente Paralelas

Aula 40: Tolerância a Falhas

Celso L. Mendes, Stephan Stephany

LAC / INPE

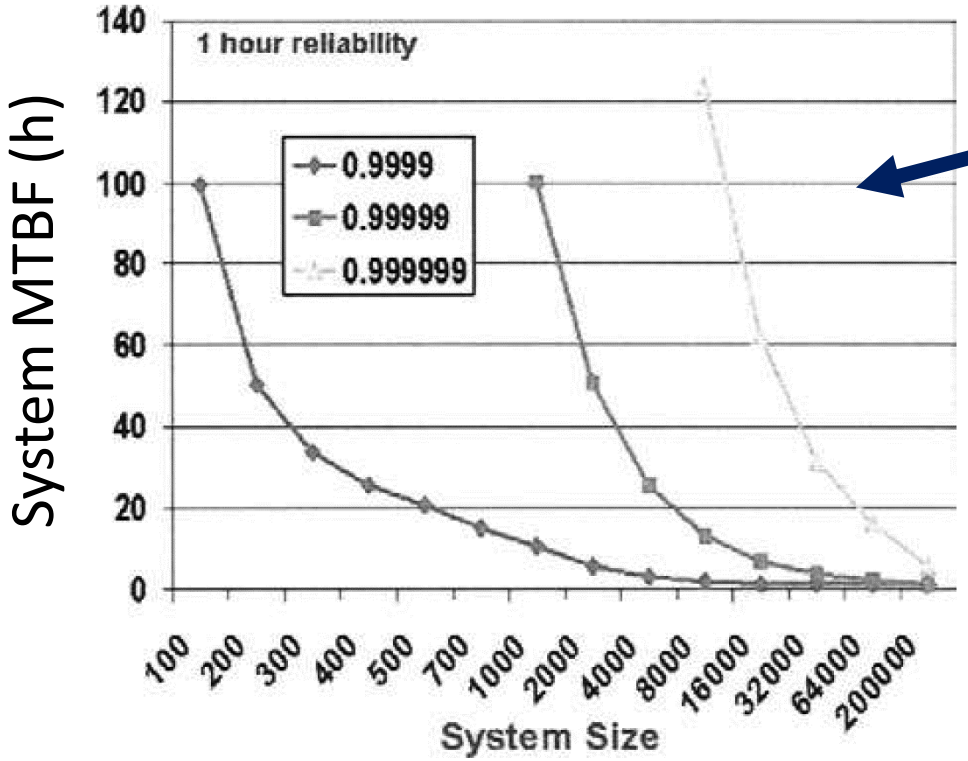
Emails: celso.mendes@inpe.br, stephan.stephany@inpe.br

Motivação

- Sistemas massivamente paralelos atuais
  - Milhares/milhões de processadores
  - Embora a probabilidade de falhas em *um* processador seja baixa, a probabilidade de alguma falha num job grande pode não ser baixa
  - Em geral, basta que um processador falhe para que todo o job seja interrompido

Motivação

- MTBF: *Mean-Time Between Failures*



Assumindo três níveis de confiabilidade de componentes

0.9999: probabilidade do componente **não** falhar em uma hora

Ref: Reed, Lu, Mendes: *Big Systems and Big Reliability Challenges* Parallel Computing, 2004

Motivação

- Origem de falhas em sistemas reais:
  - Falhas de hardware
    - Falhas de processadores - (figura anterior)
    - Falhas em outros components (discos, rede, etc)
  - Falhas de software
    - Erros de programação
    - Erros devido a configurações inapropriadas
    - Erros devido a operação inadequada
  - Falhas ambientais
    - Quedas de energia
    - Esgotamento de recursos (ex: espaço em disco)

Solução Típica

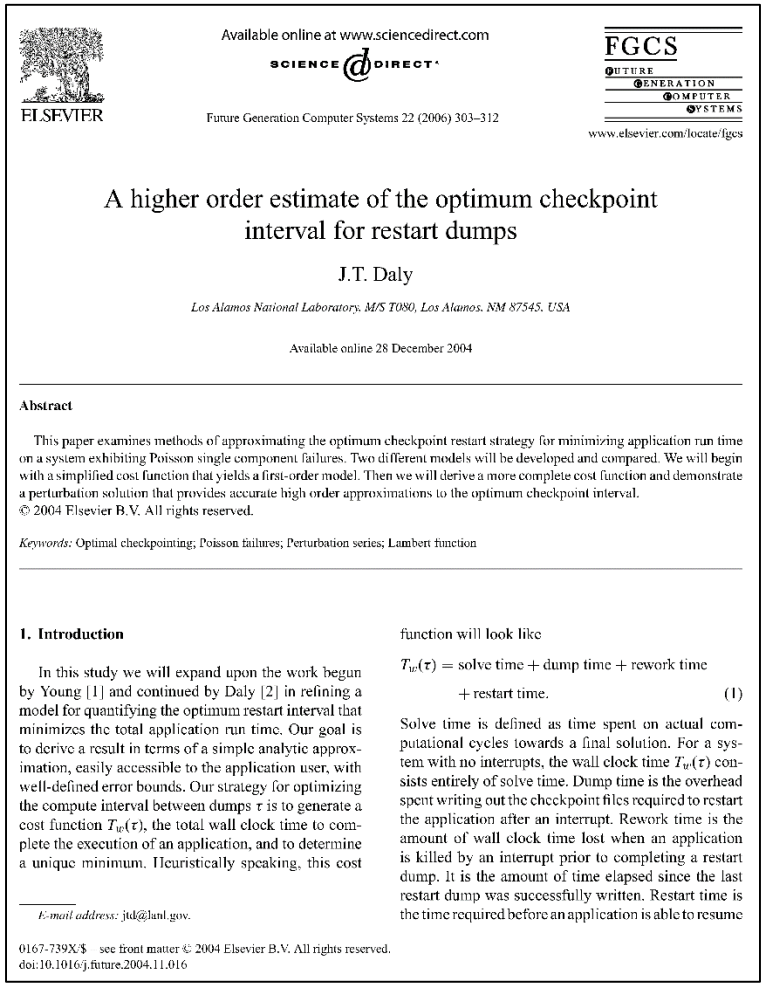
- Checkpoint/Restart:
  - Idéia básica: salvar (tipicamente em disco) o estado da computação periodicamente, num *checkpoint*
  - Em caso de falhas:
    a) Parar a execução
    b) Restaurar dados do *checkpoint* mais recente
    c) Reiniciar a execução
  - Overheads implícitos:
    - Tempo gasto para realizar checkpoints periódicos
    - Tempo de execução perdido após uma falha

Checkpoint/Restart

- Detalhes práticos
  - Normalmente trabalha-se com dois conjuntos de checkpoints, para evitar perda dos dados caso ocorra uma falha durante o processo de escrita do checkpoint
  - Problema: qual a frequência ideal de checkpoint?
    a) Frequência alta:
       - muito tempo é perdido com o overhead de realizer o checkpoint
    b) Frequência baixa:
       - muito tempo de execução poderá ser perdido quando houver uma falha
       - Execução pode não avançar caso as falhas sejam frequentes

Frequência de Checkpoint

- Referência básica:
  - Daly: *A high order estimate of the optimum checkpoint interval for restart dumps*, FGCS, 2004
    - Calcula o interval ideal para se fazer checkpoint e minimizar o tempo de execução do programa
    - Refina resultados de trabalhos anteriores



Available online at www.sciencedirect.com
SCIENCE DIRECT
Future Generation Computer Systems 22 (2006) 303–312
FGCS FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/fgcs

A higher order estimate of the optimum checkpoint interval for restart dumps

J.T. Daly

Los Alamos National Laboratory, M/S T080, Los Alamos, NM 87545, USA

Available online 28 December 2004

Abstract

This paper examines methods of approximating the optimum checkpoint restart strategy for minimizing application run time on a system exhibiting Poisson single component failures. Two different models will be developed and compared. We will begin with a simplified cost function that yields a first-order model. Then we will derive a more complete cost function and demonstrate a perturbation solution that provides accurate high order approximations to the optimum checkpoint interval.

Keywords: Optimal checkpointing; Poisson failures; Perturbation series; Lambert function

1. Introduction

In this study we will expand upon the work begun by Young [1] and continued by Daly [2] in refining a model for quantifying the optimum restart interval that minimizes the total application run time. Our goal is to derive a result in terms of a simple analytic approximation, easily accessible to the application user, with well-defined error bounds. Our strategy for optimizing the compute interval between dumps τ is to generate a cost function $T_w(\tau)$, the total wall clock time to complete the execution of an application, and to determine a unique minimum. Heuristically speaking, this cost function will look like

$$T_w(\tau) = \text{solve time} + \text{dump time} + \text{rework time} + \text{restart time}. \quad (1)$$

Solve time is defined as time spent on actual computational cycles towards a final solution. For a system with no interrupts, the wall clock time $T_w(\tau)$ consists entirely of solve time. Dump time is the overhead spent writing out the checkpoint files required to restart the application after an interrupt. Rework time is the amount of wall clock time lost when an application is killed by an interrupt prior to completing a restart dump. It is the amount of time elapsed since the last restart dump was successfully written. Restart time is the time required before an application is able to resume

E-mail address: jtd@lanl.gov.

0167-739X/$ see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2004.11.016

Modelagem Analítica

- Tempo de execução com checkpoint/restart:

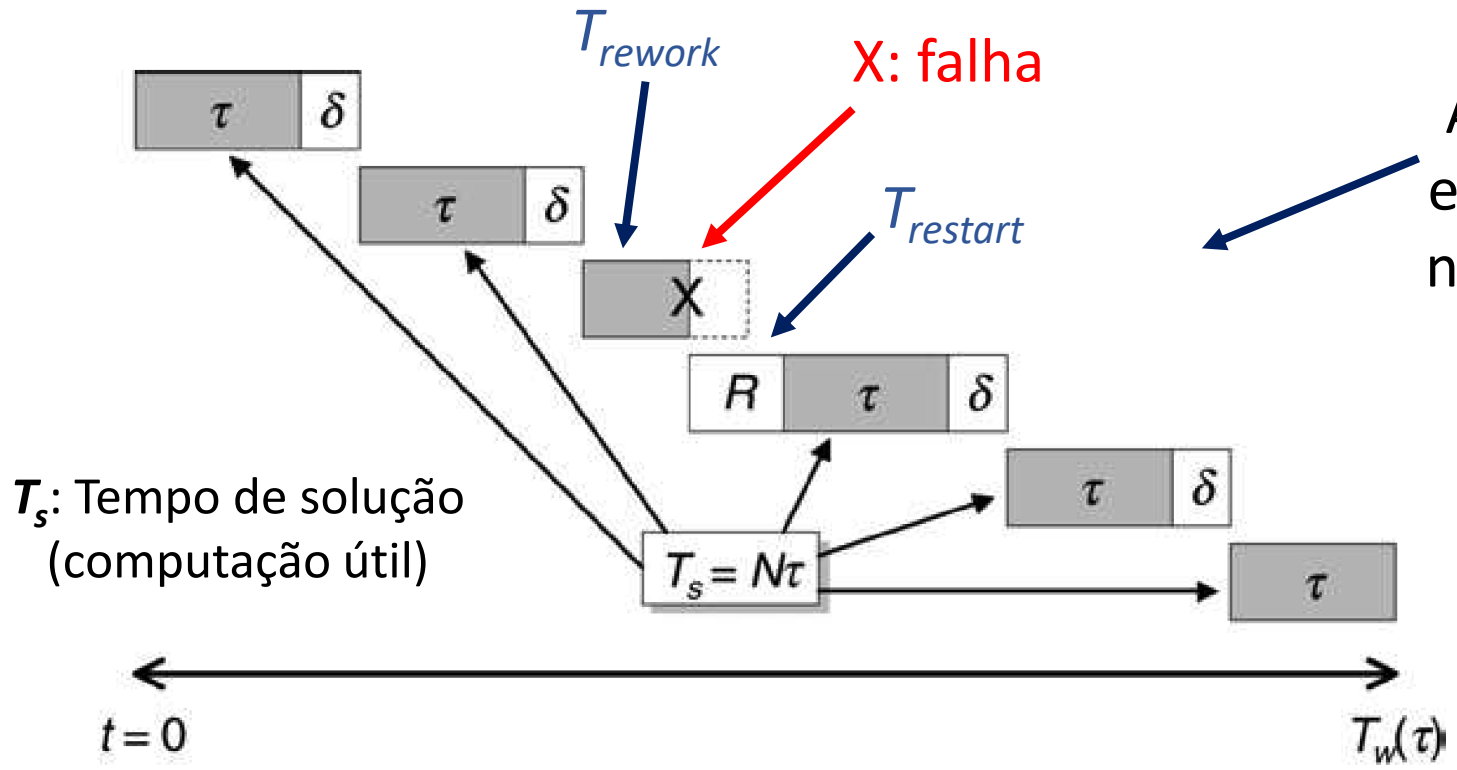
  $T_w(\tau) = T_{solve} + T_{dump} + T_{rework} + T_{restart}$ onde:

  - T_w: Tempo total de execução (*wall time*)
  - T_{solve}: tempo para solução do problema em si
  - T_{dump}: tempo gasto escrevendo arquivos de checkpoint
  - T_{rework}: tempo de execução perdido antes de uma falha
  - $T_{restart}$: tempo gasto para reiniciar execução após uma falha
  - τ : tempo de computação entre checkpoints sucessivos

  Objetivo: encontrar τ tal que $T_w(\tau)$ seja mínimo

Modelagem Analítica

- Modelo de execução com checkpoint/restart:



Modelagem Analítica

- Resultado inicial, equivalente ao de Young:
  - $\tau_{opt} = [2\delta(M+R)]^{1/2}$ para $M \gg \tau+\delta$ onde
    M é o *mean-time between failures*
  - Utiliza um modelo simplificado, de primeira ordem, para o tempo de execução
- Modelo mais sofisticado, de Daly:
  - Refina aproximações de primeira ordem no modelo
  - Permite ocorrer mais de uma falha por segmento

Modelagem Analítica

- Resultado geral, obtido por Daly:

$$\tilde{\tau}_{\text{opt}} = \begin{cases} \sqrt{2\delta M}\left[1+\frac{1}{3}\left(\frac{\delta}{2M}\right)^{1/2} + \frac{1}{9}\left(\frac{\delta}{2M}\right)\right] - \delta & \text{for } \delta < 2M, \\ M & \text{for } \delta \geq 2M. \end{cases}$$

  - τ_{opt} depende apenas de M e δ ; independe de R!
  - Notar que τ_{opt} independe também do tempo total

Resultado Prático

- Caso δ seja pequeno em relação a M:
  - Definição de "pequeno": $\delta < 2M$

$$\tilde{\tau}_{\text{opt}} = \begin{cases} \sqrt{2\delta M} - \delta & \text{for } \delta < \frac{1}{2}M, \\ M & \text{for } \delta \geq \frac{1}{2}M. \end{cases}$$

Exemplo de Uso

- Sistema com 25.000 nós, ~3 falhas de nó por dia:
  - mtbf(algum_nó) ≈ 8 horas
- Job-1: (25.000÷5)=5.000 nós

  → mtbf(job): M = 5×8 = 40 horas

  Assumindo δ=6 minutos=0,1h (tem-se $\delta < M/2$):

  $\tau_{opt}(2\times0,1\times40)^{1/2} - 0,1 \approx 2,7h = 162$ minutos
- Job-2: 25.000 nós (máquina inteira)

  → mtbf(job): M = 8 horas

  Assumindo δ=20 minutos=0,33h (ainda tem-se $\delta < M/2$):

  $\tau_{opt}(2\times0,33\times8)^{1/2} - 0,33 \approx 1,97h \approx 118$ minutos