Programa de Pós-Graduação em Engenharia de Automação e Sistemas
Universidade Federal de Santa Catarina

Emprego do Linux Perf no Monitoramento da Execução de Aplicações

Projeto de Dissertação de Mestrado
Orientador: Rômulo Silva de Oliveira
Co-Orientador: Cristian Koliver

1. Contexto

Os sistemas submetidos a requisitos temporais, também chamados de sistemas de tempo real, possuem requisitos que variam muito com relação ao tamanho, complexidade e criticalidade. Por exemplo, enquanto um mp3 player é relativamente simples, aplicações aviônicas e automotivas são extremamente complexas, possuindo dezenas de controladores eletrônicos interligados através de redes e barramentos de dados dedicadas.

No contexto da automação industrial, são muitas as possibilidades (ou necessidades) de empregar sistemas com requisitos de tempo real. Exemplos são os sistemas de controle embutidos em equipamentos industriais, os sistemas de supervisão e controle de células de manufatura e os sistemas responsáveis pela supervisão e controle de plantas industriais completas.

Na literatura os sistemas de tempo real são, em geral, classificados conforme a criticalidade dos seus requisitos temporais. Enquanto o não atendimento dos requisitos temporais nos sistemas de tempo real não críticos (soft real-time systems) resulta apenas na redução da utilidade destes sistemas (ou qualidade do serviço prestado), o não atendimento de um requisito temporal em sistemas de tempo real críticos (hard real-time systems) pode ter como resultado a perda de vidas humanas.

Nos sistemas em geral (que não são do tipo tempo real), a única preocupação é com a qualidade dos resultados. Embora uma execução rápida seja desejável, a abordagem é sempre do tipo "fazer o trabalho usando o tempo que for necessário". Sistemas tempo real possuem uma abordagem diferente, pois o tempo é limitado. É preciso garantir que será possível atender aos prazos, geralmente impostos pelo ambiente do sistema. Logo, a preocupação é “fazer o trabalho usando o tempo disponível”. Esta preocupação tem como consequência um problema básico encontrado na construção de sistemas tempo real (críticos ou não): a alocação e o escalonamento das tarefas nos recursos computacionais disponíveis. Existe uma dificuldade intrínseca em compatibilizar dois objetivos fundamentais: garantir que os resultados serão produzidos no momento desejado e dotar o sistema de flexibilidade para adaptar-se a um ambiente dinâmico e, assim, aumentar sua utilidade.

Em um extremo (em geral associado a aplicações de tempo real críticas) existem soluções de escalonamento que supõem um conjunto fixo de tarefas a serem executadas. Estas soluções reservam recursos para o pior caso e são capazes de garantir que todas as tarefas serão concluídas no momento correto. Entretanto, aplicações construídas desta forma resultam em sistemas pouco flexíveis e na subutilização dos recursos computacionais. No outro extremo temos as soluções de escalonamento que não garantem o comportamento temporal da aplicação. Tarefas são escalonadas na medida do possível. Embora os recursos computacionais sejam plenamente utilizados e o sistema resultante seja bastante flexível, a falta de uma garantia prévia para o seu comportamento temporal inviabiliza este tipo de solução para muitas classes de aplicações. A dificuldade de escalonar tarefas

com requisitos de tempo real é bastante conhecida, constituindo uma área de pesquisa intensa atualmente.

1.1 Verificação em Tempo de Execução

A verificação em tempo de execução está preocupada com o monitoramento e análise da execução de software e sistemas de hardware. Técnicas de verificação de tempo de execução são cruciais para a exatidão do sistema, confiabilidade e robustez. Elas são significativamente mais poderosas e versáteis do que testes convencionais, e mais práticas do que a verificação formal exaustiva. Verificação em tempo de execução pode ser usada antes da implantação, para testes, verificação e para fins de depuração, e após a implantação para garantir a confiabilidade, segurança e para a prestação de contenção de falhas e recuperação, bem como a reparação do sistema online. Temas de interesse incluem:

- Linguagens de especificação
- Instrumentação de programas
- Técnicas de construção de monitores
- Registros e re-execuções
- Controle em tempo de execução, detecção de faltas, recuperação
- Adaptação do programa
- Coleta de dados estatísticos
- Visualização da execução de programas
- Monitoração de comportamento temporal

1.2 Linux Perf

Perf é uma ferramenta profiler para sistemas baseados em Linux 2.6+ que abstrai as diferenças de hardware em medições de desempenho Linux e apresenta uma interface de linha de comando simples. Perf é baseado na interface perf_events exportado por versões recentes do kernel do Linux.

A ferramenta perf oferece um rico conjunto de comandos para coletar e analisar os dados de desempenho e de rastreamento. O uso de linha de comando é uma reminiscência do git em que há uma ferramenta genérica, perf, que implementa um conjunto de comandos: stat, registro, relatório, etc.

A ferramenta perf suporta uma lista de eventos mensuráveis. A ferramenta e a interface do kernel subjacente podem medir eventos provenientes de diferentes fontes. Por exemplo, alguns eventos são contadores de kernel puros, neste caso chamados de eventos de software. Os exemplos incluem: Context-switches, pequenas faltas.

Outra fonte de eventos é o próprio processador e sua Unidade de Monitoramento de Desempenho (PMU - Performance Monitoring Unit) do processador. Ele fornece uma lista de eventos para medir eventos micro-arquitetônicos, tais como o número de ciclos, instruções, erros de cache L1 e assim por diante. Esses eventos são chamados de eventos de hardware PMU ou eventos de hardware. Eles variam de acordo com cada tipo de processador e modelo.

2. Objetivo

Explorar a utilização da ferramenta perf em soluções para a verificação de aplicações de tempo real executando sobre o sistema operacional Linux. O objetivo é monitorar o comportamento do sistema durante a fase de testes para levantar dados estatísticos e depois, com a aplicação no campo, monitorar seu comportamento para detectar desvios, os quais podem indicar violações de segurança, falha intermitente do hardware, ou ainda condições de execução (dados de entrada) diferentes daqueles previstos durante o desenvolvimento do sistema.

A dissertação deverá incluir um amplo levantamento bibliográfico sobre as abordagens existentes na literatura para a verificação em tempo de execução. Em especial, considerar aplicações com requisitos temporais importantes. Buscar avaliar as propostas da literatura com respeito a eficácia e facilidade de uso (custo de implementação).

Espera-se ao final que o mestrando conheça os princípios da verificação em tempo de execução e da monitoração de execução, e como estes princípios impactam a execução de aplicações de tempo real.

3. Atividades

(a) Estudar os conceitos básicos da verificação em tempo de execução.

(b) Estudar o conjunto de ferramentas associados com o perf.

(c) Implementar aplicações de tempo real com o propósito de usar o perf e consolidar o entendimento do que ele pode provê e como pode ser usado.

(d) Realizar experiências com propostas da literatura relacionadas com a verificação em tempo de execução.

(e) Propor um método onde as ferramentas associadas com o perf são utilizadas para a monitoração em tempo de execução de uma aplicação de tempo real.

(f) Validar o método proposto através de estudos de caso.

(g) Redigir artigos para publicação em congressos e/ou revistas.

(h) Redigir a dissertação.

4. Bibliografia

N. C. Audsley, A. Burns, A. J. Wellings. Deadline Monotonic Scheduling Theory and Application. Control Engineering Practice, Vol. 1, No. 1, pp. 71-78, feb. 1993.

A. Burns, A. J. Wellings. Criticality and Utility in the Next Generation. The Journal of Real-Time Systems, Vol. 3, correspondence, pp. 351-354, 1991.

M. Kim, M. Viswanathan, I. Lee, H. Ben-Abdellah, S. Kannan, O. Sokolsky. Formally Specified Monitoring of Temporal Properties, Proceedings of the European Conference on Real-Time Systems, June 1999.

I. Lee, S. Kannan, M. Kim, O. Sokolsky, M. Viswanathan. Runtime Assurance Based On Formal Specifications, Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications, June 1999.

S. Savage, M. Burrows, G. Nelson, P. Sobalvarro, T. Anderson. Eraser: a Dynamic Data Race Detector for Multithreaded Programs. ACM Trans. Comput. Syst. 15(4), November 1997, pp. 391-411.

K. W. Tindell, A. Burns, A. J. Wellings. An Extendible Approach for Analyzing Fixed Priority Hard Real-Time Tasks. Real-Time Systems, pp. 133-151, 1994.