Dos sistemas distribuídos *self-aware*

Raimundo José de Araújo Macêdo[1], **Allan Edgard Silva Freitas**[1,2]

[1]Laboratório de Sistemas Distribuídos - Universidade Federal da Bahia (UFBA)

[2]Instituto Federal da Bahia (IFBA)

macedo@ufba.br, allan@ifba.edu.br

Abstract. *Self-aware distributed systems run algorithms that adjust its operation mode, according to the perceived execution context. That context is built from monitoring of the local state of distributed system components, whose characteristics can vary during execution time. That allows us to build more flexible systems adapted to current complex scenarios of distributed computing.*

Resumo. *Sistemas distribuídos self-aware se caracterizam por algoritmos que ajustam o seu modo de execução de acordo com o contexto de execução percebido. Este contexto é construído pela observação do estado local dos seus componentes, cujas características podem variar ao longo do tempo. Isto permite construir algoritmos adaptáveis aos atuais e complexos ambientes distribuídos.*

1. Introdução

A crescente diversidade, heterogeneidade e onipresença das redes e dispositivos computacionais tornam os cenários de composição de sistemas distribuídos complexos, como, por exemplo, computação em nuvens federada [Rochwerger et al. 2009] ou *Cyber-Physical Systems*[Lee 2008]. Nestes cenários, diferentes componentes podem possuir características distintas e tais características evoluírem ao longo da execução, como, por exemplo, o comportamento temporal (i.e. entre síncrono e assíncrono).

Em cenários como estes que as características dos componentes do sistema variam ao longo da execução, uma infraestrutura de monitoramento permite equipar os algoritmos em execução da percepção do contexto, de modo a permitir que estes adaptem seu comportamento aos requisitos existentes no ambiente.

Para representar tais ambientes, diferentes modelos de sistemas foram propostos. Em [Gorender et al. 2007], um modelo de sistema distribuído híbrido e dinâmico é apresentado. Neste o uso de um mecanismo subjacente de provimento e monitoramento da qualidade de serviço, permite (re)negociar níveis de serviço fim-a-fim entre processos (i.e. assegurando limites temporais em canais de comunicação) e perceber a qualidade de serviço existente, obtendo informação que possibilita o ajuste do desenho de algoritmos, como o consenso distribuído, que se adaptam de acordo com a percepção do ambiente. Em [Macêdo and Freitas 2009, Macêdo and Freitas 2010], generalizamos o modelo e foi apresentado um algoritmo de comunicação em grupo capaz de lidar com esses ambientes híbridos e dinâmicos.

Esta possibilidade de percepção do comportamento dos componentes do sistema, extraindo um contexto para a aplicação, que permite a esta alterar a execução em face

das características de seus componentes, torna o sistema distribuído auto-cônscio (*self-aware*). Este artigo apresenta uma discussão sobre sistemas distribuídos *self-aware*, ilustrando cenários de interesse.

2. Modelo de Sistema

Um sistema distribuído $DS(\Pi, \chi)$ é formado por um conjunto $\Pi = \{p_1, p_2, ..., p_n\}$ de processos, e um conjunto $\chi = \{(p_i, p_j), \forall p_i, p_j \in \Pi\}$ de canais de comunicação. O estado do sistema DS expressa o comportamento de seus componentes, canais e processos, em um dado instante de tempo k; e pode ser dado por um vetor de estado $U[k]$. Se o sistema for híbrido e dinâmico, seus componentes podem possuir comportamentos distintos, que podem variam ao longo do tempo de execução.

A percepção deste conjunto de características dos componentes de DS por p_i é realizada por meio de um conjunto de estimadores, baseado em monitoramento do estado local de canais e processos, compondo um estimador do estado global $\widehat{U}[k]$.

Para ser útil às aplicações de sistemas distribuídos, $\widehat{U}[k]$ deve ser construído de modo a produzir uma estado global consistente de DS[Chandy and Lamport 1985]. Este é utilizado para extrair uma semântica segura do estado para a computação. Esta semântica percebida por p_i é denominada contexto σ_i do ambiente percebido. Este contexto deve expressar como as características dos componentes do sistema evoluem ao longo da execução.

O contexto $\sigma_i[k+1]$ percebido no instante de tempo discreto $k+1$ é uma função do contexto $\sigma_i[k]$ no instante imediatamente anterior k e da informação $\widehat{U}[k]$ de p_i sobre o conjunto de estimadores em k, onde σ^0 é o contexto inicial da execução:

$$\sigma_i[k+1] = f(\sigma_i[k], \widehat{U}[k]); \sigma_i(0) = \sigma_i^0 \tag{1}$$

Em um sistema *self-aware*, mapeamos o contexto $\sigma_i[k]$ obtido de modo que os algoritmos adaptam seu comportamento para otimizar seu desempenho ao ambiente atual.

3. Da implementação de sistemas *self-aware*

Considere o modelo de sistema HA [Gorender et al. 2007], este utiliza da observação (estimadores) de qualidade de serviço temporal de processos e canais e de mecanismos de detecção de defeitos. O comportamento temporal dos componentes do sistema pode ser mapeado em dois comportamentos $timely$ e $untimely$: um dado processo é *timely* se existe valor máximo conhecido (digamos, ϕ) para a execução de passos de computação em p_i. Da mesma forma, um canal c_i é *timely* se uma mensagem é transmitida em c_i dentro de um limite de tempo limitado e conhecido, digamos δ. Do contrário, processos e canais são *untimely*. δ e ϕ são parâmetros do sistema de execução e garantidos por mecanismos de sistemas operacionais e redes de tempo real.

Por meio de mecanismos de monitoramento da qualidade de serviço da infraestrutura subjacente aos componentes do sistema distribuído, podemos obter um estimador $\widehat{QoS}$ da função QoS que mapeia o comportamento temporal de cada componente. De forma similar, assumindo, por exemplo, a hipótese de falhas de processos por *crash*, uma função detector de defeitos pode mapear o estado de um processo entre $alive$ e $suspected$.

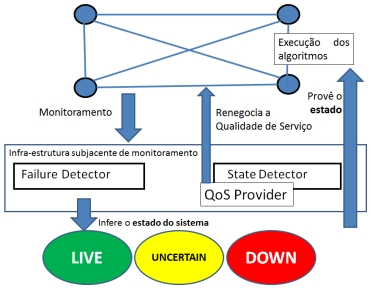

Figura 1. Exemplo de sistema distribuído *self-aware*

Utilizando mecanismos de monitoramento, como, por exemplo, *heartbeats*, implementamos um detector de defeitos $\widehat{FD}$, o qual associa a percepção de funcionamento, entre $alive$ e $suspected$ a cada um dos processos do sistema.

Desta forma os estimadores $\widehat{QoS}$ e $\widehat{FD}$ mapeiam os processos em uma semântica que exprime subconjuntos de processos para os quais assumimos a certeza do funcionamento ($live$) ou da falha ($down$) ou a impossibilidade de identificar o funcionamento correto ($uncertain$).

O conjunto de regras $R0 - R6$ do modelo HA define as transições da função f que determinam σ_i na forma de sub-conjuntos $live_i$, $down_i$ e $uncertain_i$ para cada processo p_i, conforme a observação do ambiente e a percepção anterior destes sub-conjuntos.

Dentre as regras, destaca-se a hipótese que apenas degradação e não melhorias na qualidade de serviço de canais de comunicação são percebidas pelos processos durante a execução do algoritmo distribuído. Ou seja, o conjunto $live_i$ mantém-se ou é monotonicamente decrescente em tempo de execução (processos anteriormente em $live_i$ podem ser percebidos em $uncertain_i$ por p_i se a qualidade de serviço decai). Este conjunto de regras de transição de estados fornece uma semântica conservadora, mas segura, a qual permite a execução do consenso distribuído, adaptando-se ao longo da execução o quórum aos processos em $live$ e a maioria de processos em $uncertain$.

Desta forma, utilizando este modelo de sistema, em [Gorender et al. 2007] é apresentado um consenso adaptativo, no qual o quórum necessário para a realização do consenso se adapta a quantidade atual de componentes síncronas de acordo com os conjuntos (*life, uncertain e down*). Em [Macêdo and Freitas 2010] é implementado um protocolo de comunicação em grupo, baseado em blocos causais [Macêdo et al. 1993], em que cada processo utiliza desta informação de contexto, dada pelos conjuntos, e da *QoS* percebida dos canais de comunicação, de modo a adaptar o comportamento, determinando condições de entrega de mensagens baseado em limites temporais quando estes existirem, otimizando a execução em face da existência de componentes síncronos.

A figura 1 ilustra o exemplo de sistema distribuído *self-aware*, com uma estrutura de provisionamento de recursos para obter qualidade de serviço (*QoS provider*) e um monitor desta QoS (*State Detector*) – estes, em conjunto com um mecanismo de monitoramento de processos ativos baseado em *heartbeats*, detector de defeitos (*failure detector*), permitem obter a informação dos conjuntos *life*, *uncertain* e *down* utilizados para o ajuste da operação do algoritmo distribuído.

Considere um cenário distinto: uma rede veicular *ad-hoc* (*VANET*). Pode-se com-

por um protocolo de roteamento baseado em *clusters*, de modo a prover escalabilidade: nós se agrupam por proximidade formando um *cluster*, cujo líder é responsável pelo encaminhamento das mensagens de seu grupo.

Podemos utilizar um estimador de localização geográfica $\widehat{loc_i}$ associado ao nó de cada processo p_i, baseada em um sensor GPS, de modo a propor para cada processo p_i um contexto de localizações $locations_i$ dos nós do sistema.

Cada nó de uma *VANET self-aware* pode utilizar esse conjunto $locations_i$ para se auto-organizarem em *clusters*, englobando os nós de uma mesma região, e determinando o líder do *cluster* que encaminhará mensagens aos demais *clusters*.

4. Considerações Finais

A percepção de como as características de componentes de um sistema distribuído possam variar ao longo do tempo nos permite prover adaptação dos algoritmos ao longo da execução definindo sistemas *self-aware*. Um sistema distribuído *self-aware* é cônscio do seu contexto de execução, mapeado a partir do estado local de seus componentes, provendo informação deste contexto para os processos que o compõem.

O uso deste contexto, como o dado pela QoS dos componentes ou pela localização de nós em *VANETs*, permite a algoritmos em execução adaptarem o seu comportamento. Tais sistemas se diferenciam dos convencionais por uma infraestrutura subjacente de monitoramento, baseado em estimadores do estado local dos componentes, que provê uma camada de contexto para o uso da aplicação distribuída.

Referências

Chandy, K. and Lamport, L. (1985). Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems*, 3(1):63–75.

Gorender, S., Macêdo, R. J. A., and Raynal, M. (2007). An adaptive programming model for fault-tolerant distributed computing. *IEEE Transactions on Dependable and Secure Computing*, 4(1):18–31.

Lee, E. (2008). Cyber physical systems: Design challenges. In *Proc. of the 11th IEEE Int. Symp. on Object Oriented Real-Time Distributed Computing*, pages 363–369.

Macêdo, R. J. A., Ezhilchelvan, P., and Shrivastava, S. K. (1993). Modeling group communication using causal blocks. In *Proc. of the 5th European Workshop on Dependable Computing*.

Macêdo, R. J. A. and Freitas, A. E. S. (2009). A generic group communication approach for hybrid distributed systems. In *Proc. of the 9th IFIP Int. Conf. on Distributed Applications and Interoperable Systems*, volume 5523 of *LNCS*, pages 102–115.

Macêdo, R. J. A. and Freitas, A. E. S. (2010). Group communication for self-aware distributed systems. In *Proc. of the Braz. Symp. on Computer Networks and Distributed Systems*, pages 915–928.

Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente, I. M., Montero, R., Wolfsthal, Y., Elmroth, E., Caceres, J., Ben-Yehuda, M., Emmerich, W., and Galan, F. (2009). The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4):4:1–4:11.