Sistemas Distribuídos - Estudo artigo Prefetchers

1. **Introdução**

Ao definirmos a necessidade de desenvolvimento de um sistema levantamos as necessidades de negócio em requisitos funcionais a serem atendidos e, de um modo limitado por ainda estar no início, requisitos não funcionais para definir a plataforma na qual será desenvolvido o sistema, infraestrutura disponível, linguagem de programação, bancos de dados, dentre outros. Uma visão de sistemas de computadores em rede pode ser definida e para o caso de sistemas distribuídos temos uma Visão Mais Ampla onde os sistemas se estendem através da infraestrutura de computadores em rede disponibilizada. A extensão permitirá que os recursos do sistema fiquem próximos de onde são necessários. Essa visão relaciona-se mais frequentemente a sistemas distribuídos com seus processos e recursos distribuídos suficientemente entre vários computadores.

Dentro da arquitetura desenhamos uma solução atendendo as perspectivas que definem o sistema como um sistema distribuído.

No refinamento do escopo temos então a definição dos componentes desta arquitetura que irão compor os serviços dentro do negócio a ser atendido e na arquitetura que atenda a demanda estimada de uso de recursos, requisitos não funcionais, como por exemplo, “o sistema deve ter um carregamento rápido para o usuário” e com este requisito dentre outros implementamos componentes e processos que não estarão exatamente ligados ao negócio.

1. **Projeto do sistema**

*“1.2 Design goals*

*Just because it is possible to build distributed systems does not necessarily mean that it is a good idea.*”

Um sistema distribuído que valha a pena ser desenvolvido deve atender a alguns objetivos que são imprescindíveis:

* 1. Compartilhamento de recursos: distribuir e otimizar o acesso a recursos remotos para usuários e outros sistemas.

Uma forma de alcançar este objetivo poderia ser a utilização de princípios SOLID na definição da arquitetura dos componentes e sua implementação. Componentes bem formados utilizando:

S - Single Responsability na formação de suas classes onde deve ser especializada em um único assunto e possuir apenas uma responsabilidade dentro do software, ou seja, a classe deve ter uma única tarefa ou ação para executar.

O - A abertura a novas extensões dos componentes e um fechamento para alterações nas classes atuais do sistema OCP – Open-Closed Principle permitirá evoluções ao sistema e implantações de forma isolada e não perceptíveis aos usuários.

Além destes dois princípios estruturais temos outros 3 relacionados a boas práticas da implementação da arquitetura.

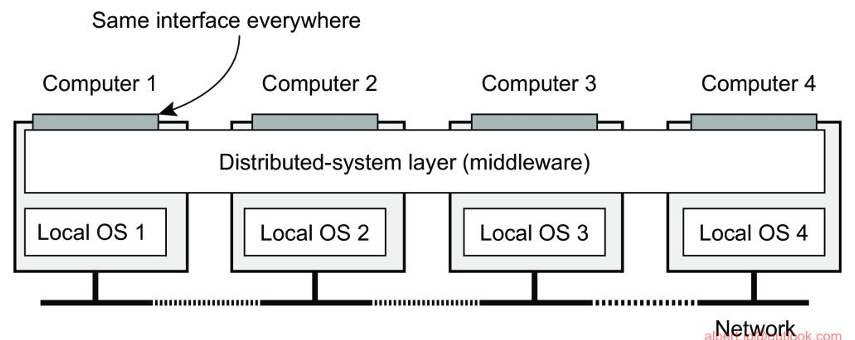
L — Liskov Substitution Principle (Princípio da substituição de Liskov)

I — Interface Segregation Principle (Princípio da Segregação da Interface)

D — Dependency Inversion Principle (Princípio da inversão da dependência)

* 1. Transparência na distribuição

Ocultar os processos e recursos distribuídos entre vários computadores torna a distribuição deste invisível a usuários finais e aplicações que irão acessar os serviços. A camada que intermedia ou faz a comunicação entre um sistema distribuído e seus usuários finais e aplicações é denominada Middleware.



* 1. Abertura

“*An open distributed system is essentially a system that offers components that can easily be used by, or integrated into other systems.*”

Pode-se alcançar flexibilidade em sistemas distribuídos abertos, seguindo os princípios que mencionei anteriormente de SOLID para que então tenhamos um sistema que seja organizado como uma coleção de componentes relativamente pequenos e facilmente substituíveis ou adaptáveis.

“*The need for changing a distributed system is often caused by a component that does not provide the optimal policy for a specific user or application*.”

Componentes bem formados são coesos, com baixo acoplamento e estarão em uma arquitetura que os disponibiliza próximos aos usuários e aplicações que necessitam de seus recursos e processos.

Neste ponto recuperamos um dos requisitos de exemplo que é atendido com a proposta ou conceito apresentado no artigo científico.

“Um processamento de retorno de resposta rápido ao utilizar os recursos e processos do sistema”. Como incluir a um sistema que praticamente já atende os requisitos funcionais novos componentes que irão agregar ao sistema uma garantia de operação sem falhas e ainda uma execução mais rápida e otimizada nas chamadas de processos ou comunicações entre componentes.

Uma visão que podemos ter de um sistema distribuídos e seus componentes é em relação as políticas e comportamentos inerentes a cada parte. O mecanismo dá o recurso a ser utilizado para uma atividade ou processamento e a política define como é utilizado este recurso. O comportamento da política de um recurso refere-se mais ao modo como o usuário ou aplicação cliente irá utilizar o recurso do que propriamente uma definição que seja própria do recurso. Isso quer dizer que quanto mais adaptáveis ou customizáveis as políticas de um mecanismo mais assertivo é o seu funcionamento e sucesso. Separar políticas e mecanismos é o melhor caminho a seguir e em contrapartida quanto maior a separação, maior a quantidade de mecanismos para garantir que os mecanismos oferecidos são apropriados.

1. **Prefetchers (Pré-buscadores)**

Uma forma de tornar o processamento de retorno de respostas dos processos mais rápido é a diminuição das latências de acesso a memórias. Definindo o termo latência vemos que em sistemas de computação e redes, latência refere-se ao atraso de tempo entre a solicitação de um evento e a resposta ou percepção do mesmo. Em outras palavras, é o tempo que um sistema leva para processar uma solicitação e retornar uma resposta. Uma forma de diminuir o tempo de retorno de uma resposta é buscá-la em um meio de processamento que trabalhem com padrões eficientes de acesso à memória para maximizar a taxa de transferência do sistema e a utilização de recursos. Esses recursos são chamados prefetchers ou pré-buscadores.

Em sistemas geograficamente dispersos, ter uma cópia por perto pode ocultar muitos dos problemas de latência de comunicação. O cache é uma forma especial de replicação, embora a distinção entre os dois seja frequentemente difícil de fazer ou até mesmo artificial. Assim como no caso da replicação, o cache resulta na criação de uma cópia de um recurso, geralmente próximo ao cliente que acessa esse recurso. No entanto, diferentemente da replicação, o cache é uma decisão tomada pelo cliente de um recurso e não pelo proprietário de um recurso. Há uma séria desvantagem no cache e na replicação que pode afetar negativamente a escalabilidade. Como agora temos várias cópias de um recurso, modificar uma cópia a torna diferente das demais. Consequentemente, o cache e a replicação levam a problemas de consistência.

Indo além da visão da consistência de informações entre as cópias de cache temos um problema que é abordado no artigo utilizado como base de pesquisa deste trabalho quanto a agressividade com que são registrados os caches através do uso de prefetchers. A agressividade da pré-busca é um parâmetro central no gerenciamento desses padrões de acesso. Embora o aumento da agressividade da pré-busca possa melhorar o desempenho de determinados aplicativos, muitas vezes corre-se o risco de causar poluição de cache e contenção de largura de banda, levando a uma degradação significativa do desempenho em outras cargas de trabalho. Embora muitos pré-buscadores existentes dependam de controladores de agressividade integrados estáticos ou simples, uma abordagem mais flexível e adaptativa baseada no feedback no nível do sistema é essencial para alcançar o desempenho ideal em ambientes de computação paralela. A abordagem flexível é relativa às políticas definidas pelos usuários e aplicativos clientes que utilizam este mecanismo mencionado anteriormente.

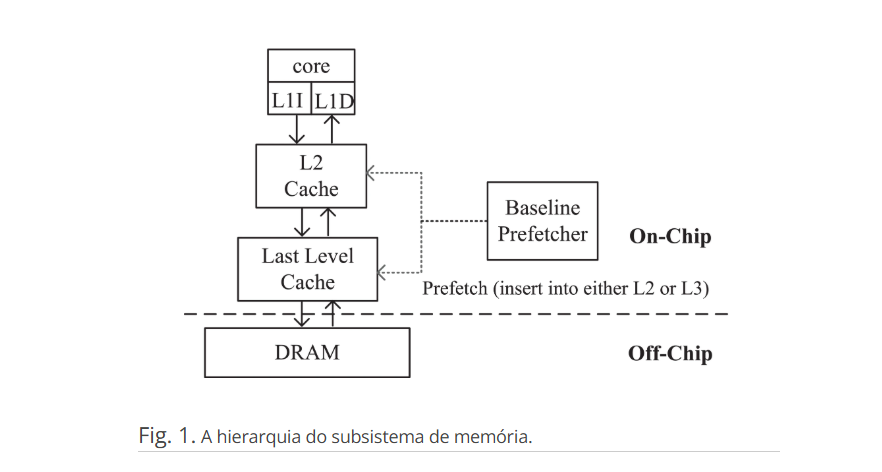
1. **Controle de Agressividade de Pré-busca Adaptativa (APAC)**

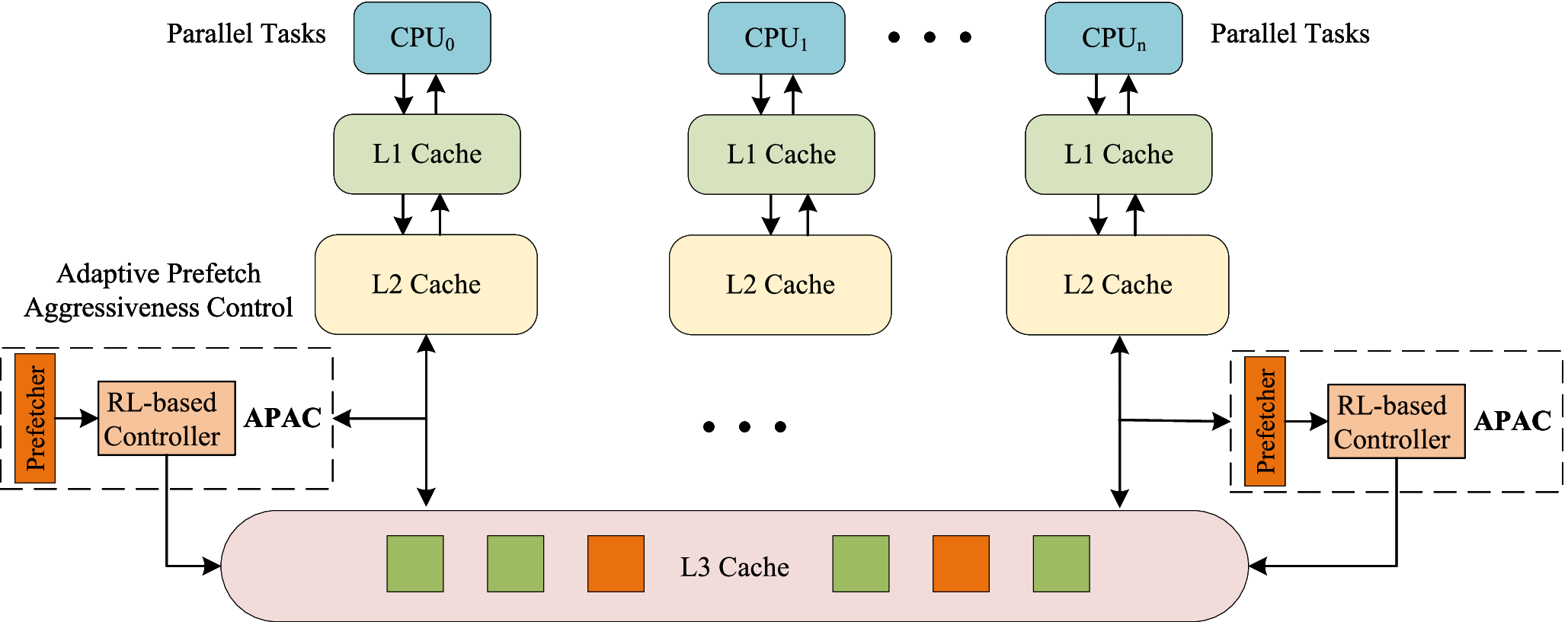
*Adaptive Prefetch Aggressiveness Control*

O artigo científico utilizado apresenta uma estrutura de Controle de Agressividade de Pré-busca Adaptativa (APAC) que aproveita o Aprendizado por Reforço (RL) para gerenciar dinamicamente a agressividade de pré-busca em arquiteturas de sistema paralelo. Reinforcement Learning é um mecanismo de controle de agressividade de pré-busca (APAC) para arquiteturas de sistemas paralelos. O controlador APAC opera como um agente RL, que otimiza a agressividade da pré-busca respondendo dinamicamente aos comentários do sistema sobre precisão de pré-busca, pontualidade e poluição do cache. O agente recebe um sinal de recompensa que reflete o impacto de cada ajuste no desempenho e na largura de banda da memória, aprendendo a adaptar sua estratégia de controle com base nas características da carga de trabalho. Essa adaptabilidade orientada por dados torna a APAC particularmente adequada para arquiteturas paralelas, onde o gerenciamento eficiente de recursos entre núcleos é essencial para dimensionar o desempenho do sistema.

Ao melhorar a utilização da largura de banda da memória, reduzir a poluição do cache e minimizar a interferência entre núcleos, o APAC melhora significativamente o desempenho de pré-busca em processadores multi-core. Esses resultados ressaltam o potencial da APAC como uma solução robusta para otimização de desempenho em arquiteturas de sistemas paralelos, onde o gerenciamento eficiente de recursos é fundamental para dimensionar ambientes de processamento modernos.

A agressividade do pré-buscador, um ajuste crucial, determina o grau em que o pré-buscador recupera especulativamente o cache linhas de níveis mais profundos da hierarquia de memória. Este parâmetro influencia diretamente desempenho do sistema, controlando o equilíbrio entre a redução da latência de acesso à memória e mitigar possíveis impactos negativos, como uso excessivo de largura de banda e poluição de cache. Embora o aumento da agressividade do pré-buscador possa melhorar o desempenho do sistema, particularmente em arquiteturas de sistemas paralelos, também pode introduzir contenção de largura de banda, cache poluição e degradação do desempenho em algumas cargas de trabalho.



* 1. **Benefícios ao utilizar o APAC**
* O APAC reduz os requisitos de armazenamento e a latência computacional usando o estado compactado e um mecanismo de recompensa de baixa sobrecarga, mantendo uma tomada de decisão eficiente em ambientes limitados por hardware.
* A APAC acelera o processo de aprendizado e reduz o tempo de exploração dinamicamente ajustando a função de recompensa e incorporando feedback no nível do sistema, permitindo ajustes efetivos de agressividade de pré-busca.
* O APAC equilibra a agressividade de pré-busca entre núcleos em sistemas multi-core por meio de uma estratégia de otimização conjunta, monitoramento da utilização da largura de banda para coordenar uso de recursos e evitar a degradação do desempenho devido à pré-busca excessiva por um único núcleo.
* Experimentos extensivos mostram que a APAC tem um desempenho melhor do que a agressividade atual dos pré-buscadores em várias cargas de trabalho e configurações de sistema. Pode efetivamente melhorar desempenho do sistema e eficiência da largura de banda.

1. **Sistema APAC proposto**

O APAC, um controlador de agressividade adaptativo alimentado por aprendizado por reforço, projetado para regular dinamicamente a agressividade do pré-buscador com base no feedback do sistema em tempo real. Adaptado para arquiteturas de sistemas paralelos, APAC aborda os desafios de ambientes multi-core onde vários núcleos competem por recursos compartilhados, como largura de banda de memória e espaço em cache. Ajustando a pré-busca degree, a APAC visa melhorar o desempenho em uma variedade de cargas de trabalho com uso intensivo de largura de banda e configurações. Essa adaptabilidade torna a APAC adequada para paralelos complexos caracterizados por núcleos e threads interdependentes.

A APAC opera em etapas de tempo de um milhão de instruções, durante as quais um coletor reúne várias métricas de eficácia de pré-busca, como poluição de cache, precisão, e uso de largura de banda, de um pré-buscador de linha de base. Essas métricas fornecem feedback crítico sobre como a pré-busca agressiva deve ser ajustada para otimizar o desempenho e eficiência de largura de banda, particularmente em ambientes paralelos onde recursos eficientes O compartilhamento é fundamental.

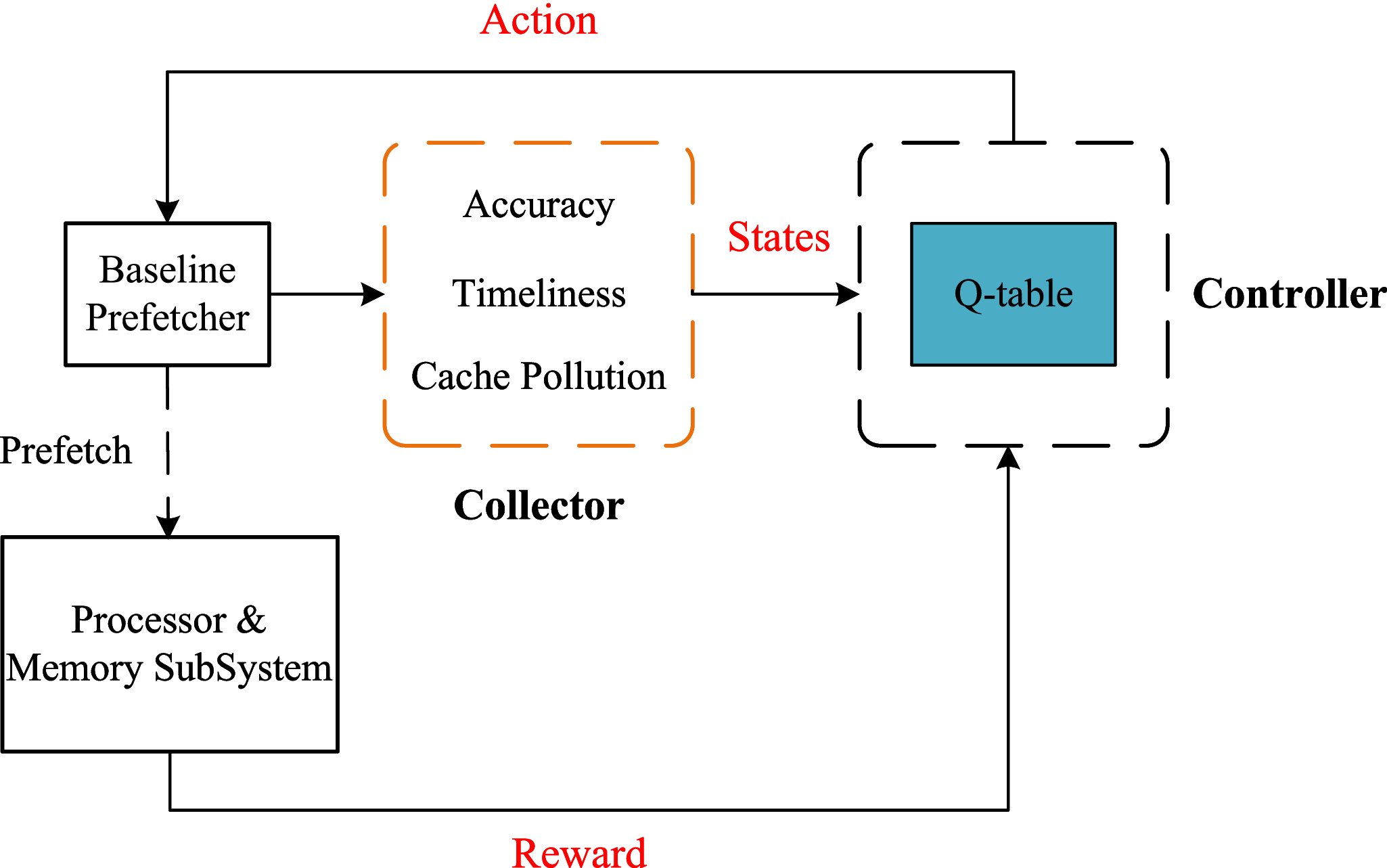


Fig. 3. Visão geral de alto nível do controlador de agressividade de pré-busca baseado em RL.

O controlador baseado em RL estabelece um par de estado-ação para cada ajuste de potencial e seleciona a ação associada ao valor Q mais alto, sinalizando o ajuste com o benefício a longo prazo mais favorável. O mecanismo de previsão de endereços da APAC é baseado em um modelo baseado em passos que rastreia padrões de acesso recorrentes; pré-buscado as linhas de cache são armazenadas no cache L2 para aumentar a disponibilidade dos dados durante o balanceamento poluição do cache. Embora o APAC seja projetado em um pré-buscador baseado em passadas, ele pode facilmente adaptar-se a outros modelos de previsão de endereços, como modelos sequenciais ou encadeados, oferecendo flexibilidade na acomodação de diferentes abordagens de pré-busca exigidas por várias demandas de carga de trabalho em sistemas multi-core.

O mecanismo de recompensa da APAC se alinha com as necessidades dos sistemas paralelos, enfatizando tanto impacto no desempenho e na largura de banda, incentivando ajustes que melhoram a utilização do cache sem consumo excessivo de recursos. Essas recompensas são usadas para atualizar os valores Q por meio do algoritmo SARSA (detalhado na Seção [III-A1](https://www.computer.org/csdl#sec3a1)), que permite à APAC aprender com cada ajuste e refinar sua política de controle ao longo do tempo. Ajustando a agressividade do pré-buscador em resposta ao sistema em tempo real feedback, a APAC garante uma utilização mais eficiente dos recursos em arquiteturas de sistemas paralelos, aprimorando assim a estabilidade geral do desempenho entre núcleos e cargas de trabalho.