Universidade Federal do Paraná — UFPR
Programa de Pós-Graduação em Informática — PPGINF
Defesa de Tese de Doutorado

Estratégias de Mapeamento de Threads para Processamento Eficiente de Consultas

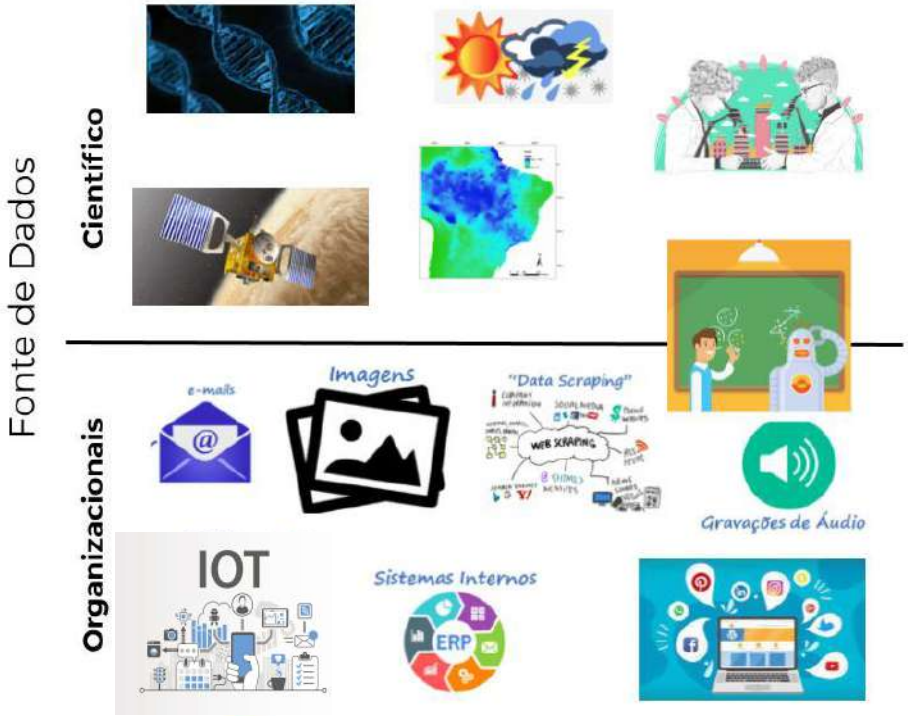
Doutoranda: Ms.Simone Dominico
Orientador: Dr.Eduardo Cunha de Almeida
Co-orientador: Dr. Marco Antônio Zanata Alves

CAPES
serrapilheira
UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

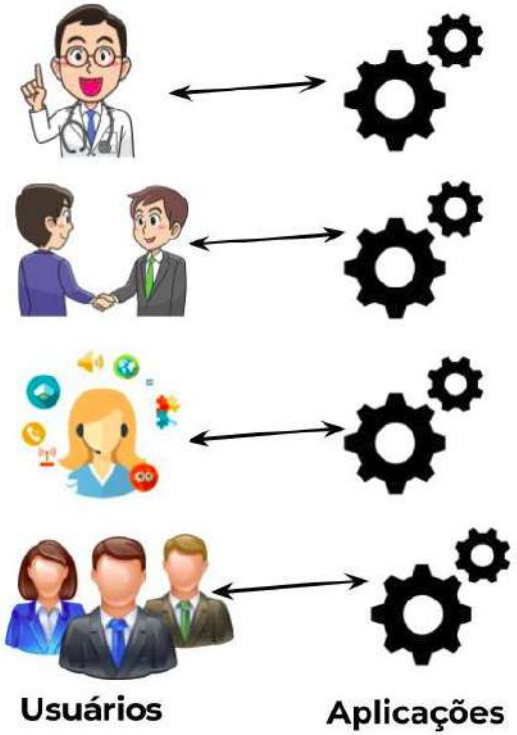
Sumário

Contexto – Dados

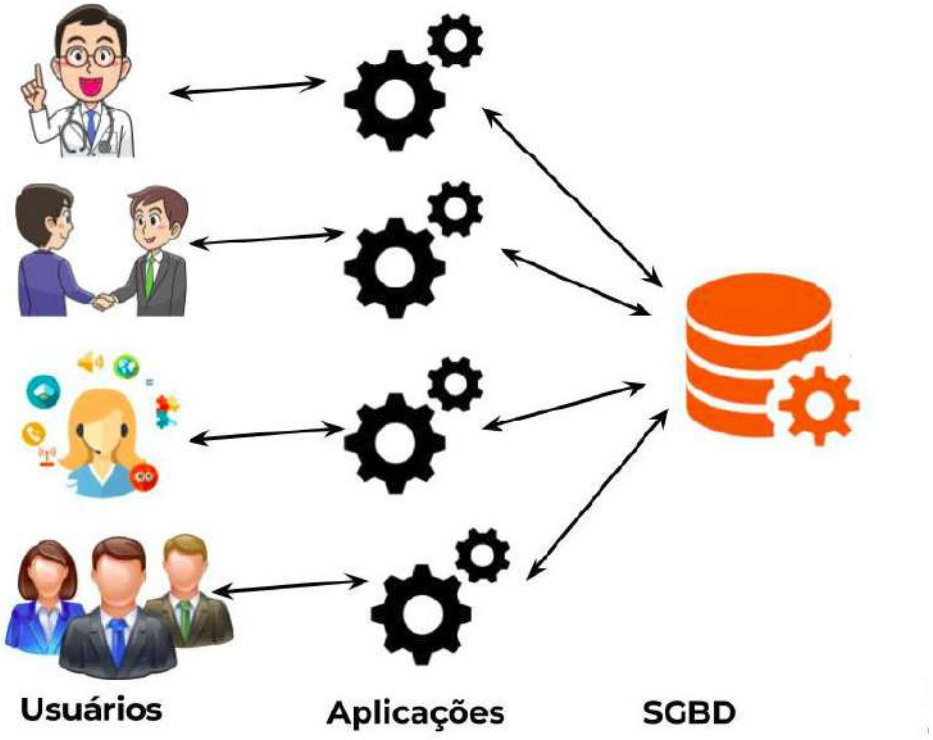
180 zettabytes para o ano 2025 [Holst, 2021].



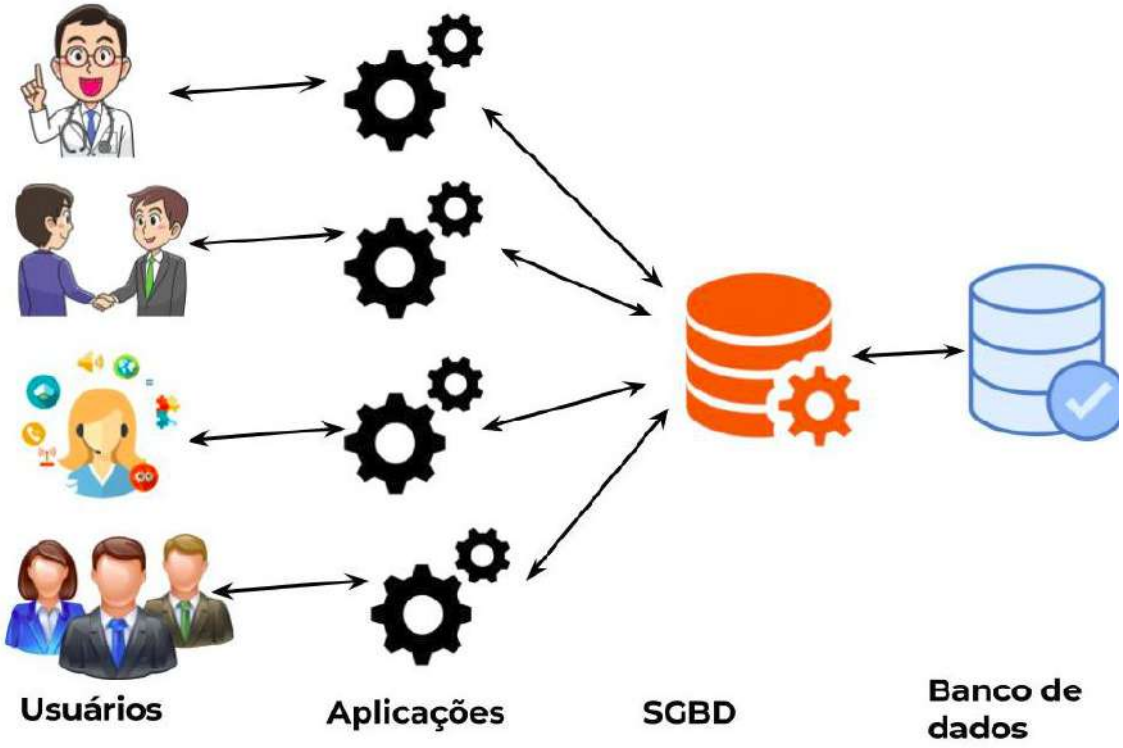
Sistema Gerenciador de Banco de Dados (SGBD)



Sistema Gerenciador de Banco de Dados (SGBD)



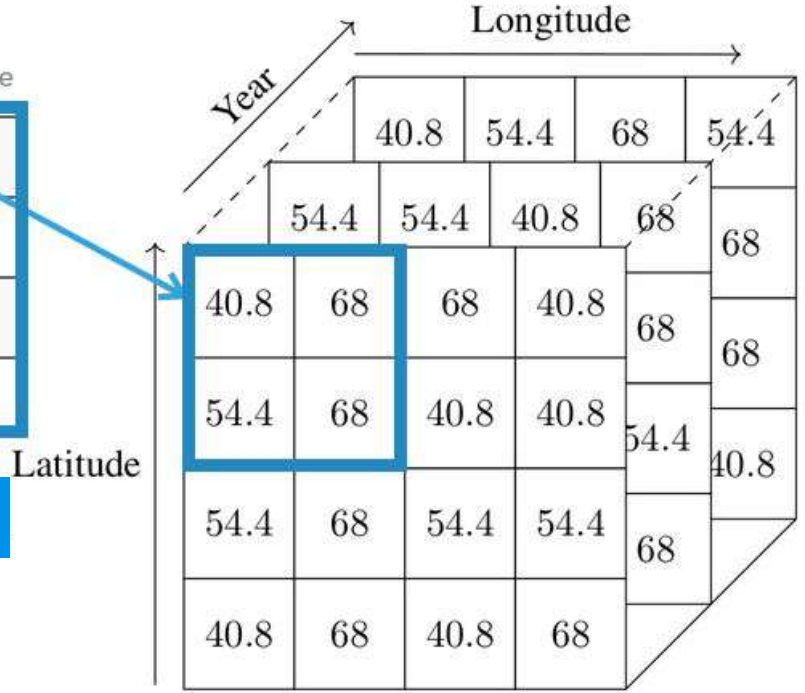
Sistema Gerenciador de Banco de Dados (SGBD)



Modelos de dados elencados na tese

Year	Latitude	Longitude	Temperature
1989	-25° 25' 40''	-49° 16' 23''	68
1989	-30° 01' 59''	-51° 13' 48''	54.4
1989	-05° 05' 21''	-42° 48' 07''	40.8
1989	-02° 31' 47''	-44° 18' 10''	54.4

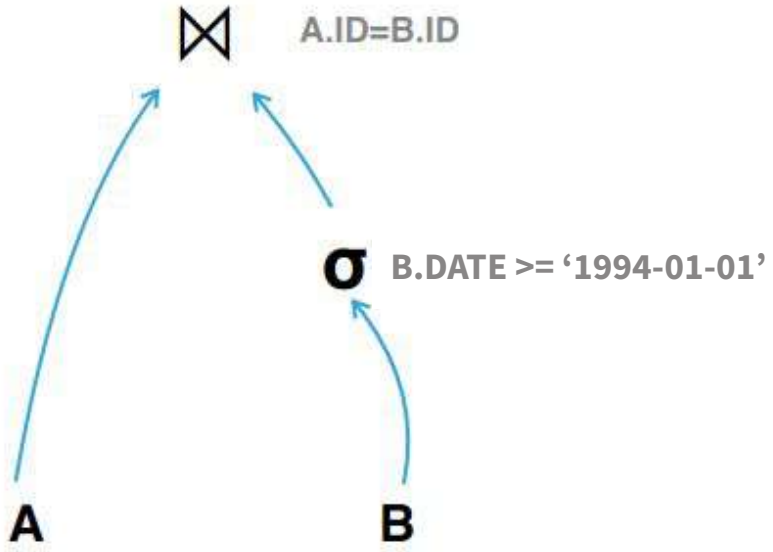
MODELO RELACIONAL



MODELO DE MATRIZ N-DIMENSIONAL

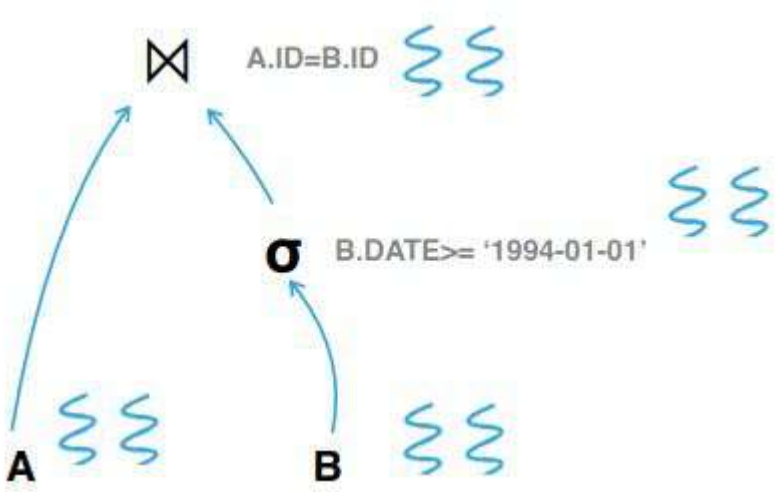
Processamento de consultas

```
SELECT *
FROM A,B
WHERE A.ID = B.ID
AND B.DATE >= ‘1994-01-01’
```

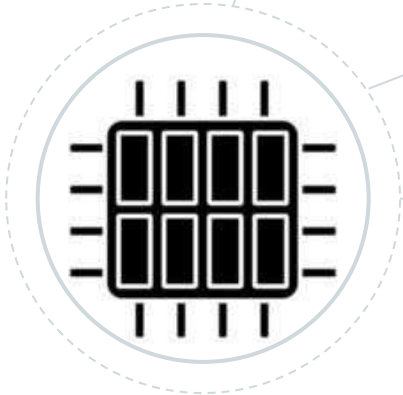


Processamento paralelo de consultas

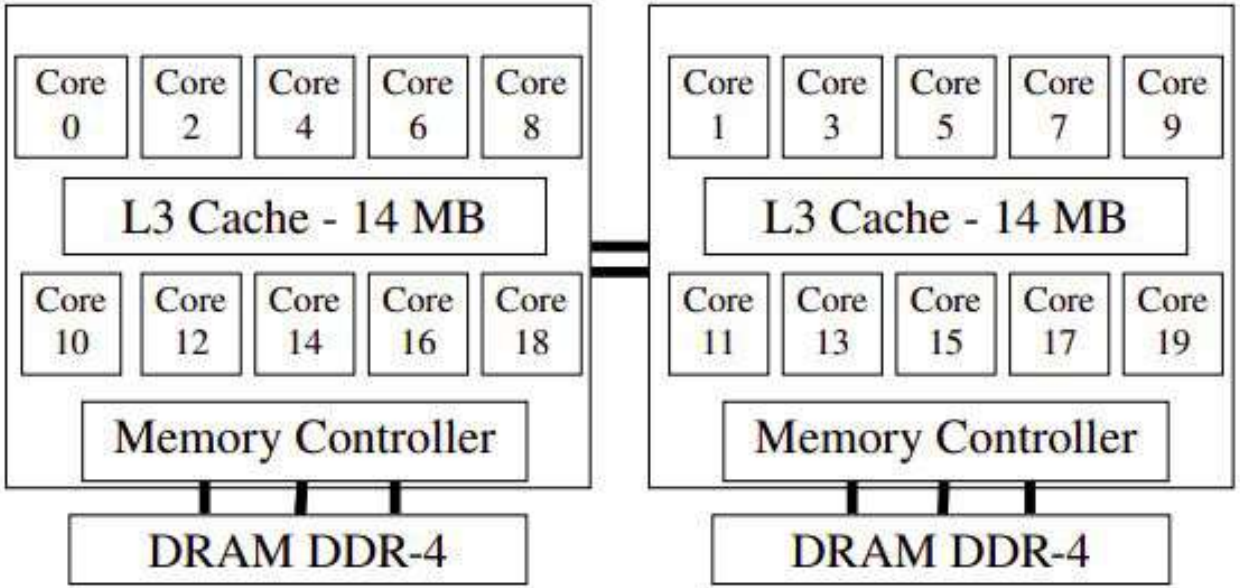
```
SELECT *
FROM A,B
WHERE A.ID = B.ID
AND B.DATE >= ‘1994-01-01’
```



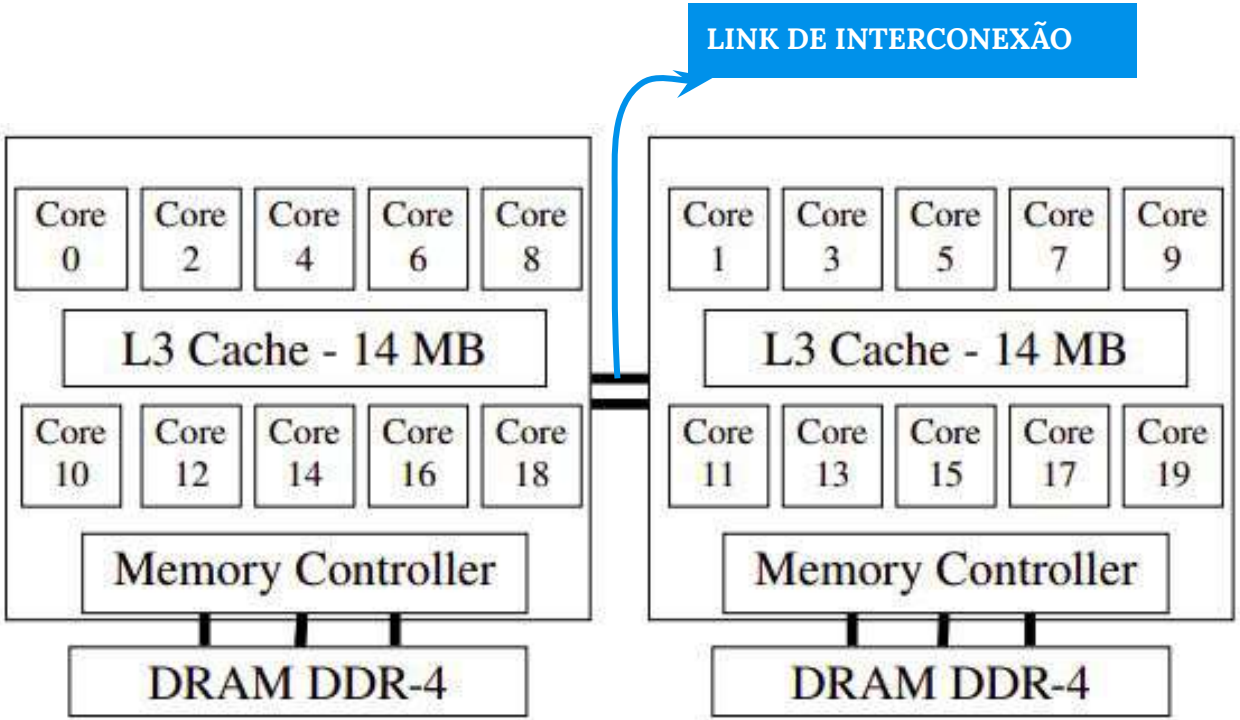
Arquiteturas multiprocessador



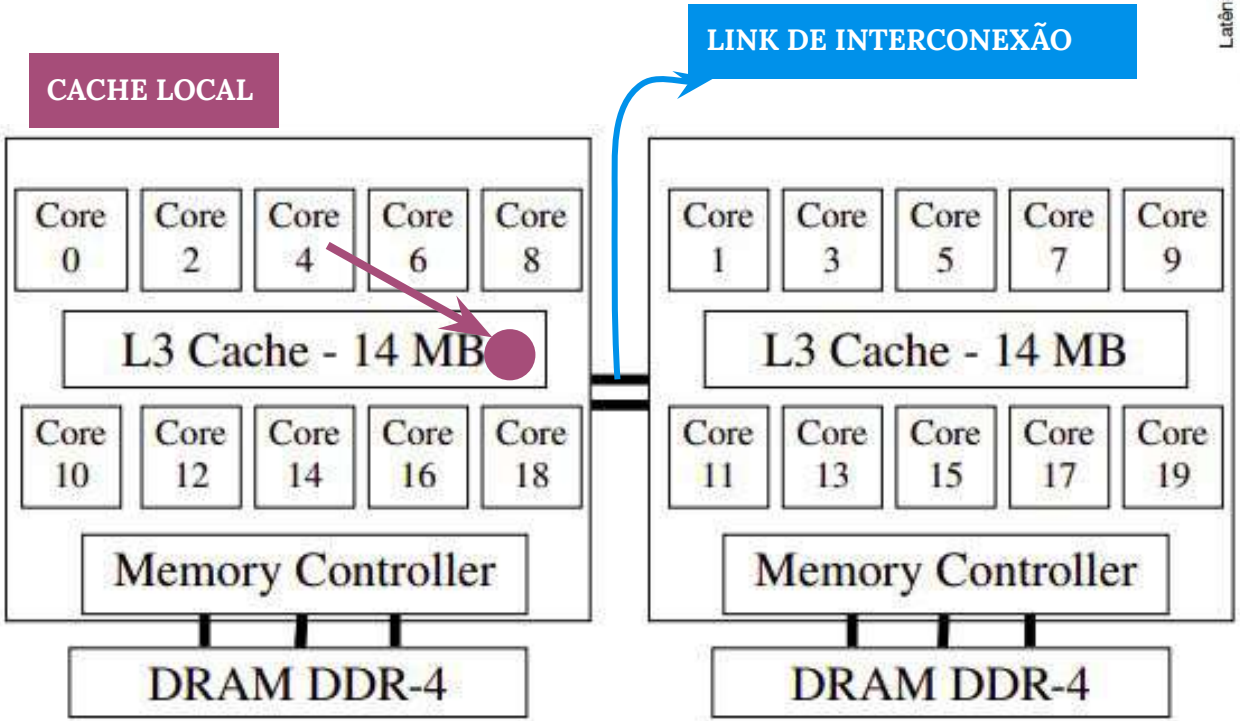
Arquitetura NUMA – *Non-Uniform Memory Access*

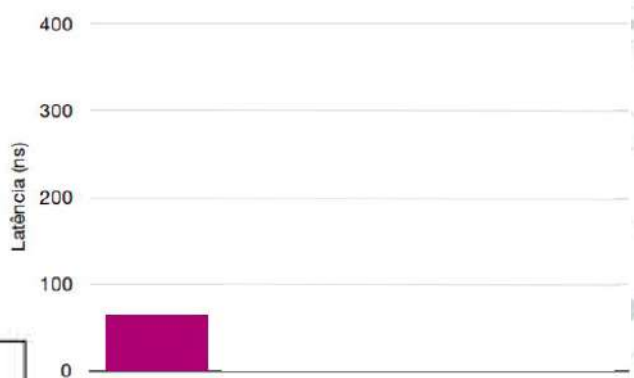


Arquitetura NUMA – *Non-Uniform Memory Access*

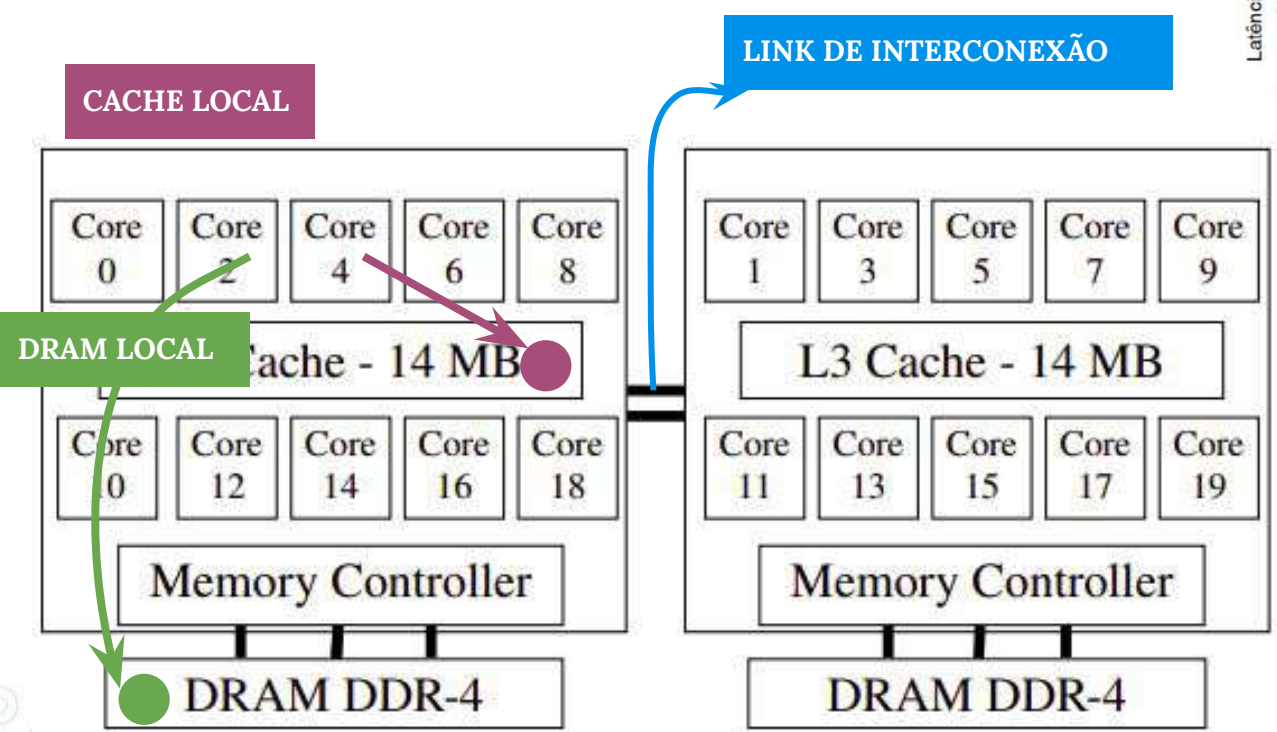


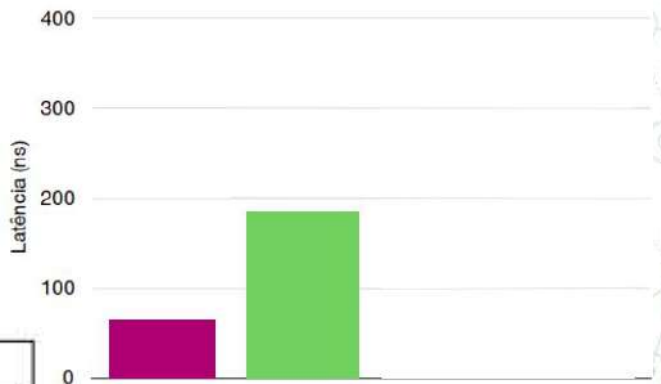
Arquitetura NUMA



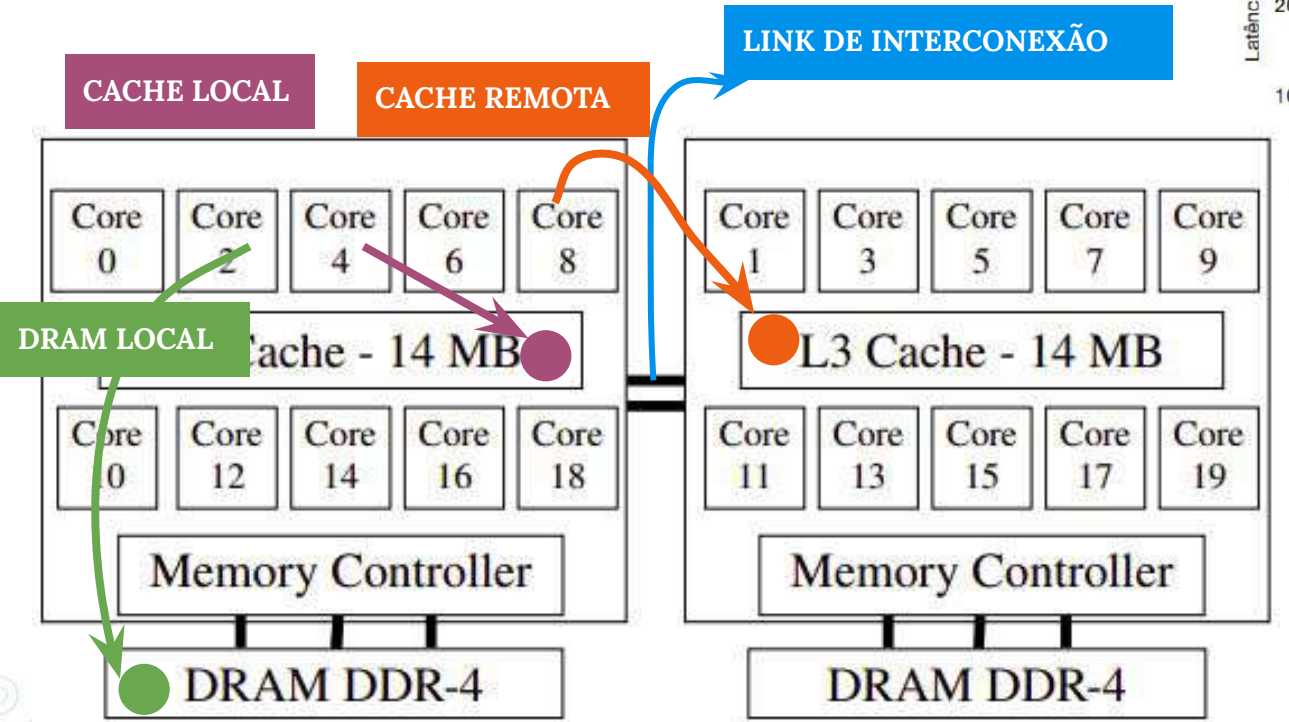


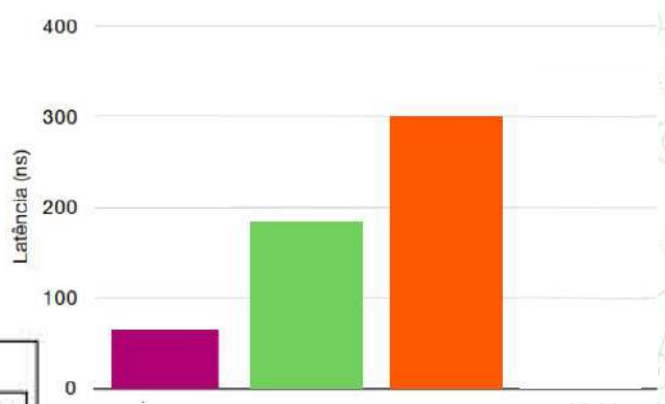
Arquitetura NUMA



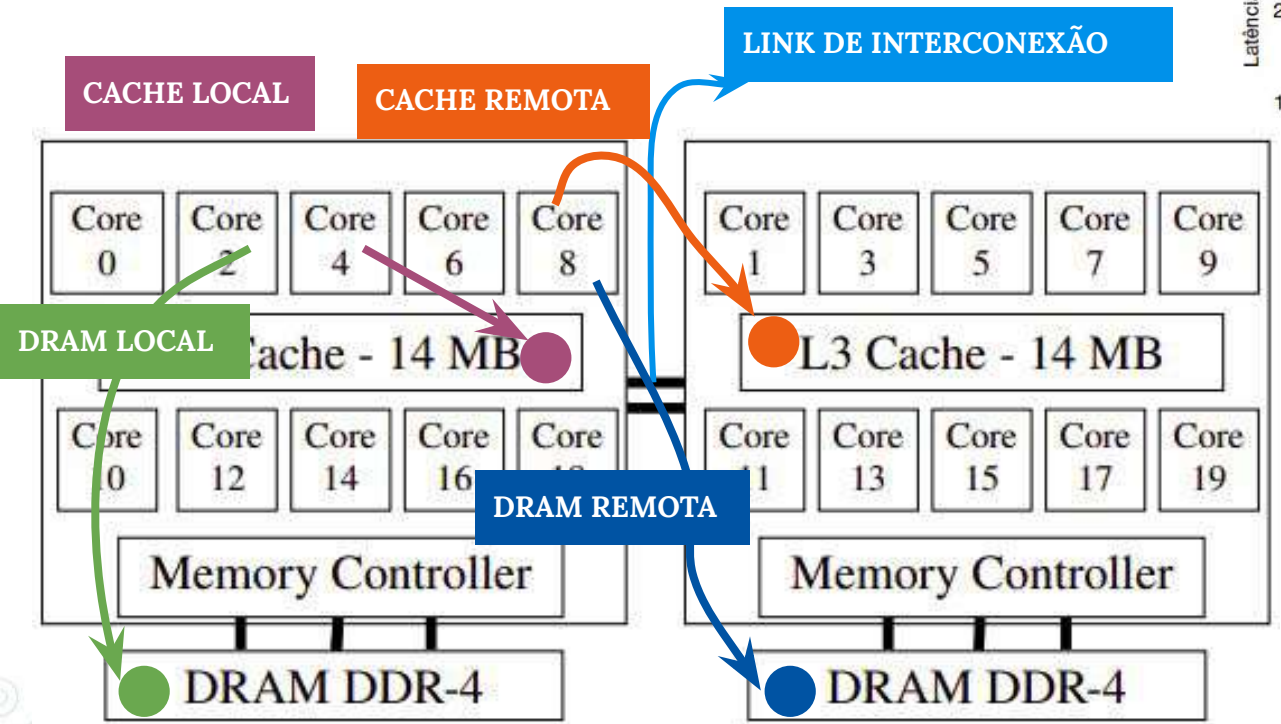


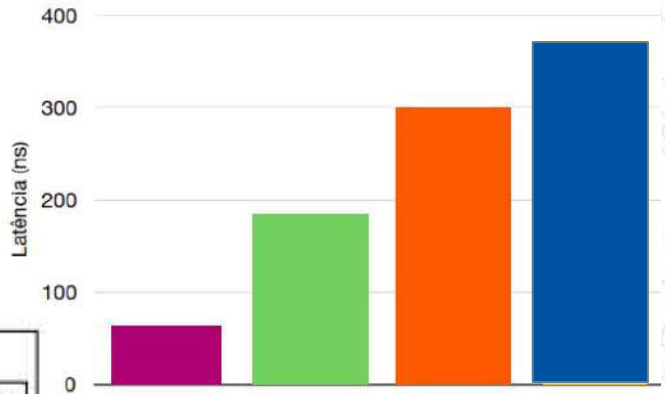
Arquitetura NUMA



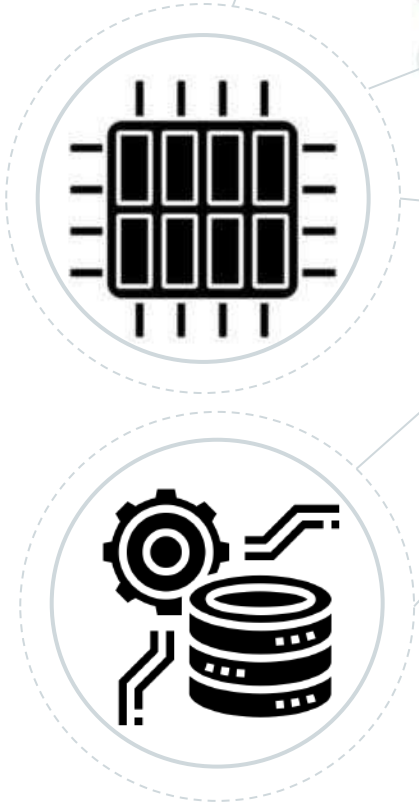


Arquitetura NUMA



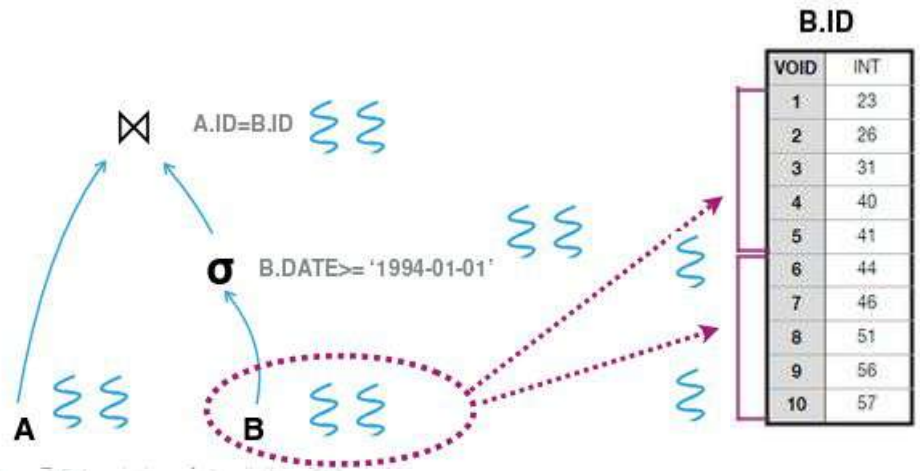


Efeito da arquitetura NUMA no processamento de consulta



Processamento de consultas relacionais

```
SELECT *
FROM A,B
WHERE A.ID=B.ID
AND B.DATE>= '1994-01-01'
```

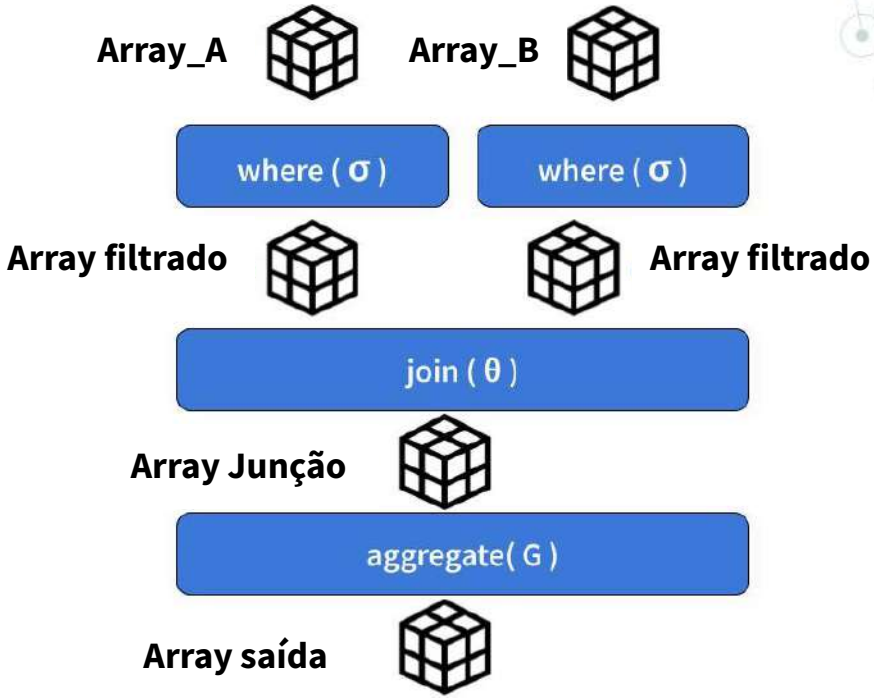


- Paralelismo é definido na etapa de planejamento e otimização de consultas.

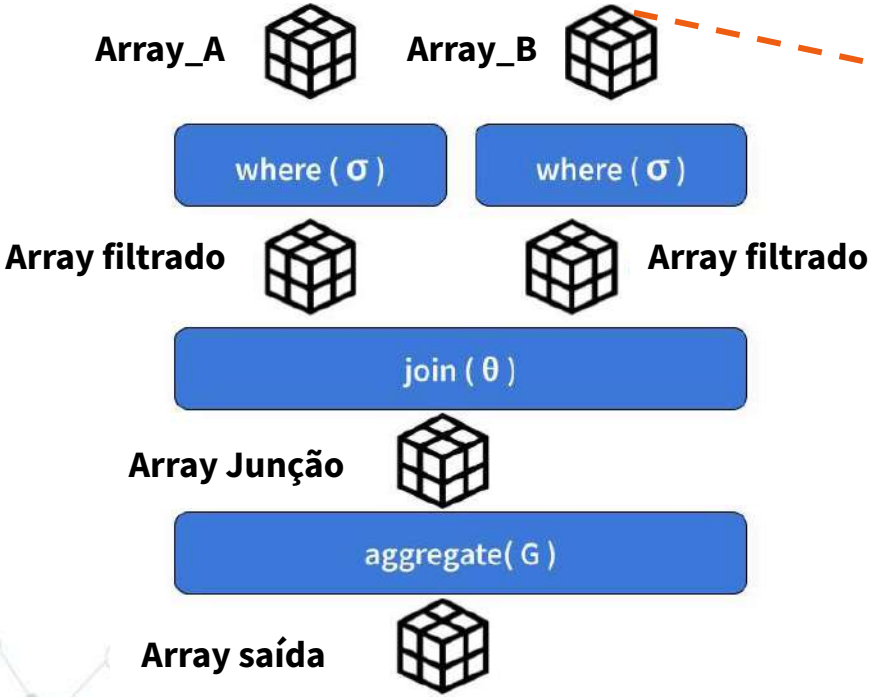
Processamento de consultas n-dimensional

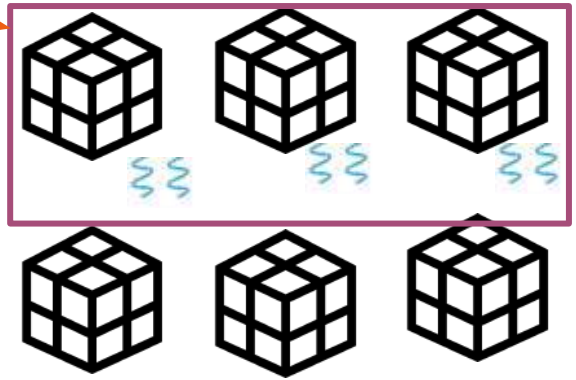
```
aggregate(
     join(
          where(array_A,
          predicate)

where(array_B,
          predicate)
     ),
     AVG
);
```



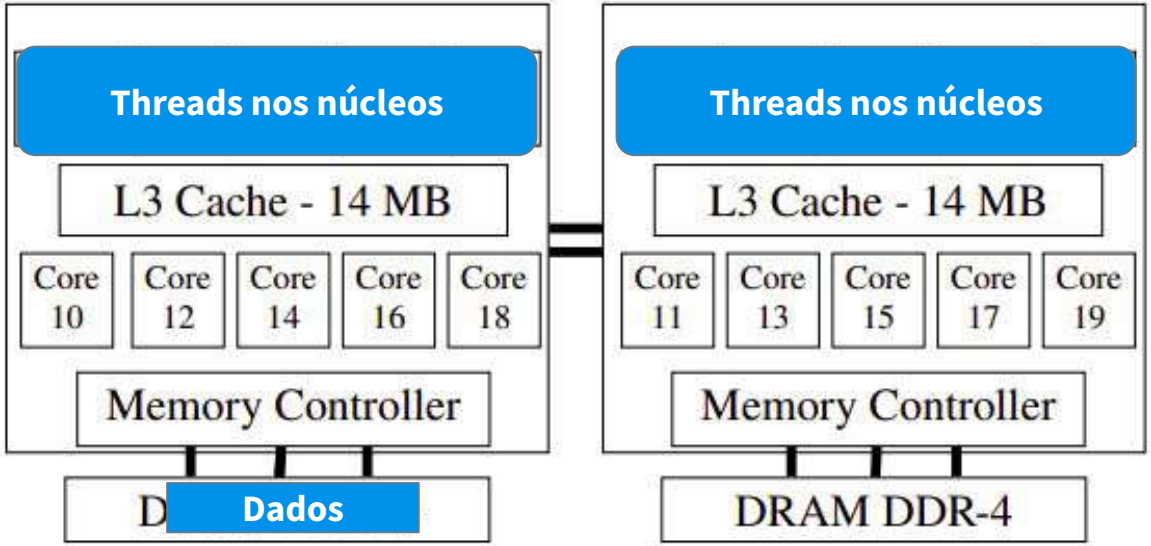
Processamento de consultas n-dimensional





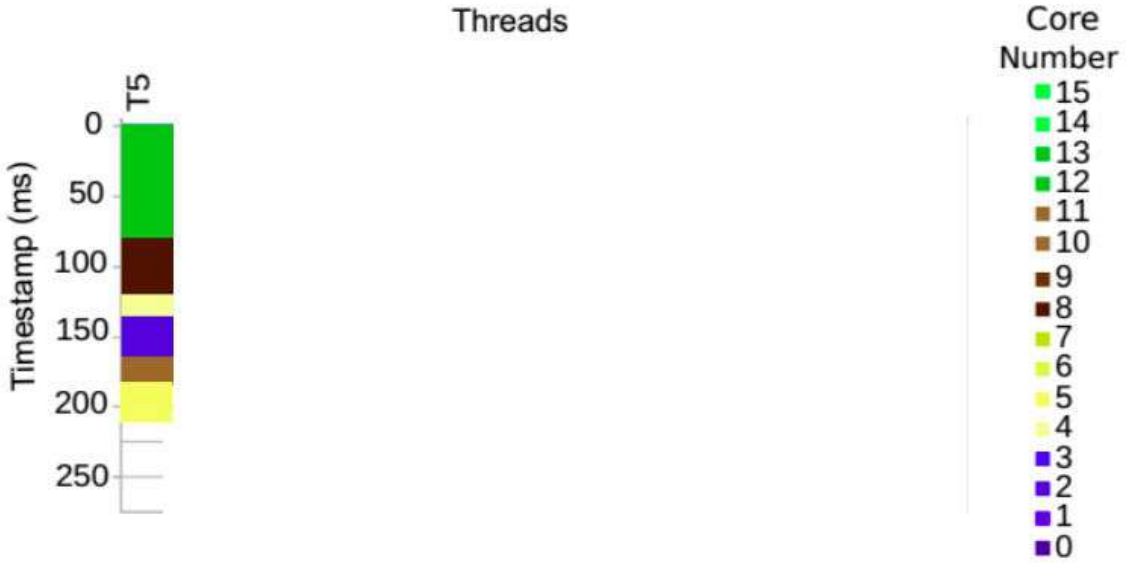
Processamento de consultas na arquitetura NUMA

- O Sistema Operacional (SO) é responsável de mapeamento
- Alocação de memória first-touch



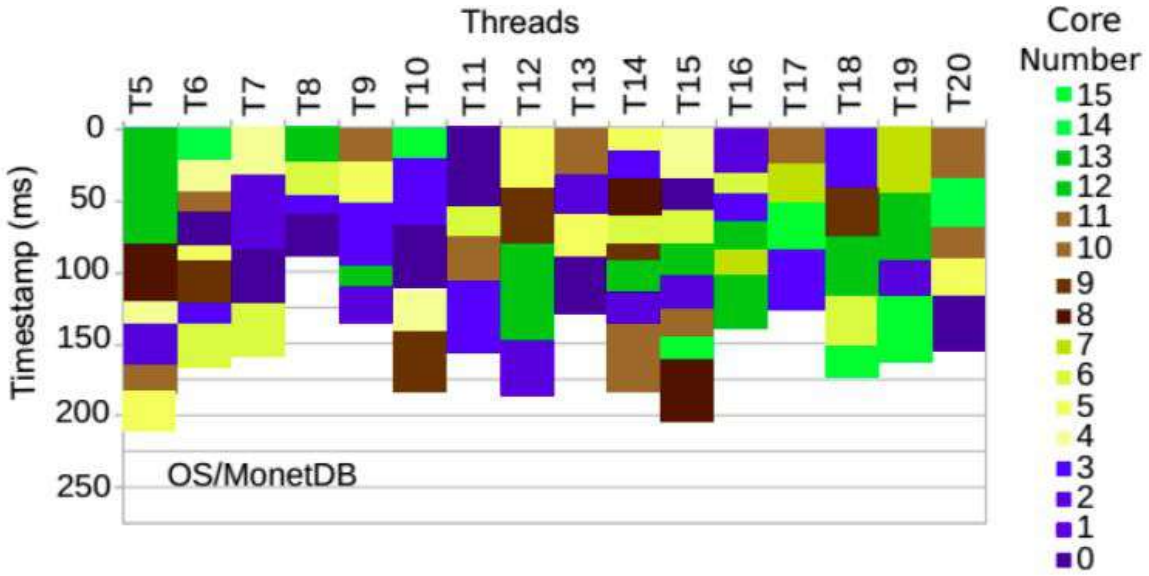
Processamento de consultas relacionais na arquitetura NUMA

- Consulta 6 — TPC-H
- Base de dados de 1GB
- 4-sockets Quad-Core AMD Opteron 8000 Series (64Gb RAM/Socket)
- SGBD: MonetDB (v11.25.5)

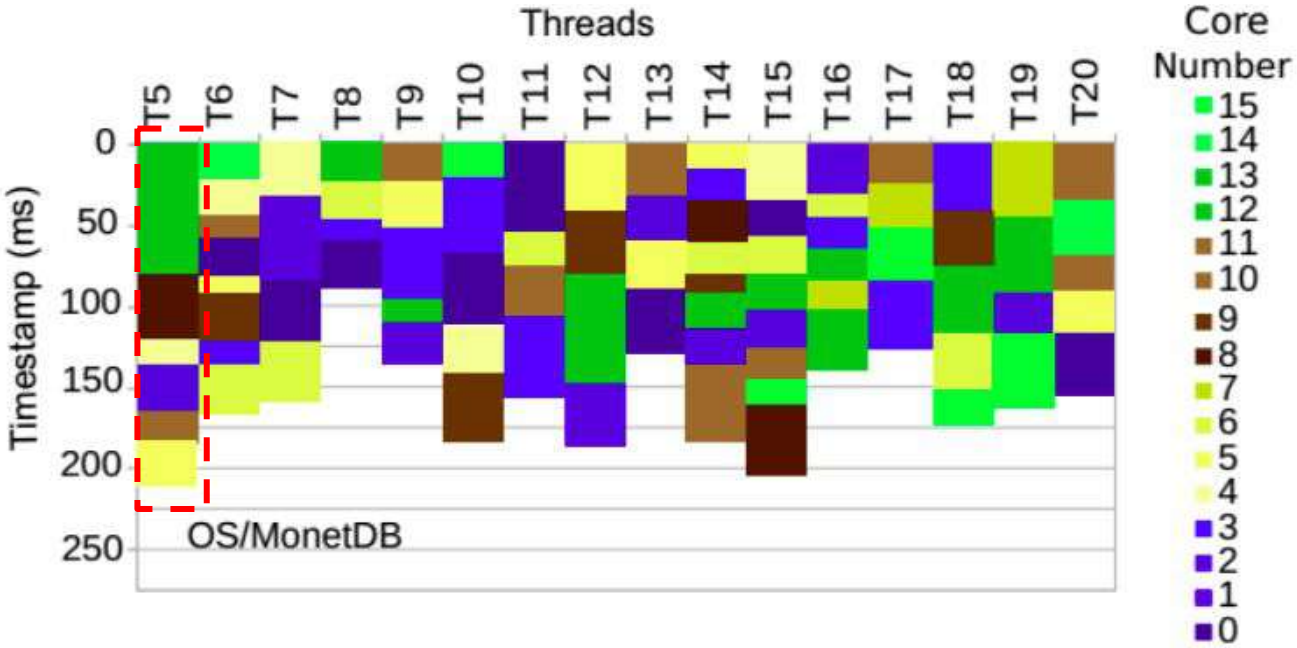


Processamento de consultas relacionais na arquitetura NUMA

- Consulta 6 — TPC-H
- Base de dados de 1GB
- 4-sockets Quad-Core AMD Opteron 8000 Series (64Gb RAM/Socket)
- SGBD: MonetDB (v11.25.5)



Processamento de consultas relacionais na arquitetura NUMA



Hipóteses

Hipóteses

1) **É possível mitigar o movimento de dados se o SO mapear as threads para um subconjunto de núcleos.**

Hipóteses

1) **É possível mitigar o movimento se o SO mapear as threads para um subconjunto de núcleos.**

2) **Podemos controlar o mapeamento de threads para determinar o nó NUMA ideal para a fixação de cada thread de uma consulta mitigando os acessos remotos.**

Perguntas de pesquisa

Perguntas de Pesquisa

Q1) Como reduzir os acessos remotos à memória garantindo que o SGBD aproveite eficientemente a arquitetura NUMA e ganhe desempenho?

Perguntas de Pesquisa

Q1) Como reduzir os acessos remotos à memória garantindo que o SGBD aproveite eficientemente a arquitetura NUMA e ganhe desempenho?

Q2) Qual o impacto da arquitetura NUMA no processamento de consultas?

Perguntas de Pesquisa

Q1) Como reduzir os acessos remotos à memória garantindo que o SGBD aproveite eficientemente a arquitetura NUMA e ganhe desempenho?

Q2) Qual o impacto da arquitetura NUMA no processamento de consultas?

Q3) Como determinar dinamicamente o melhor posicionamento das threads de processamento de consultas na arquitetura NUMA?

Trabalhos relacionados

Trabalhos relacionados — Aplicações Gerais

Trabalho Relacionado	HW	SW	Online	Dinâmico	Threads	Dados	Intrusivo	SGBD
Barrera et al. 2018		X	X		X		Aplicação	
Chiang et al. 2018		X		X	X	X	Kernel	
Serpa et al. 2018		X	X		X	X		
Chasparis et al. 2019		X	X	X	X		Aplicação	
Denoyelle et al. 2019		X			X	X		
Cruz et al. 2021	X	X	X	X	X	X	MMU	
Dominico et al. 2022		X	X	X	X		SGBD	X

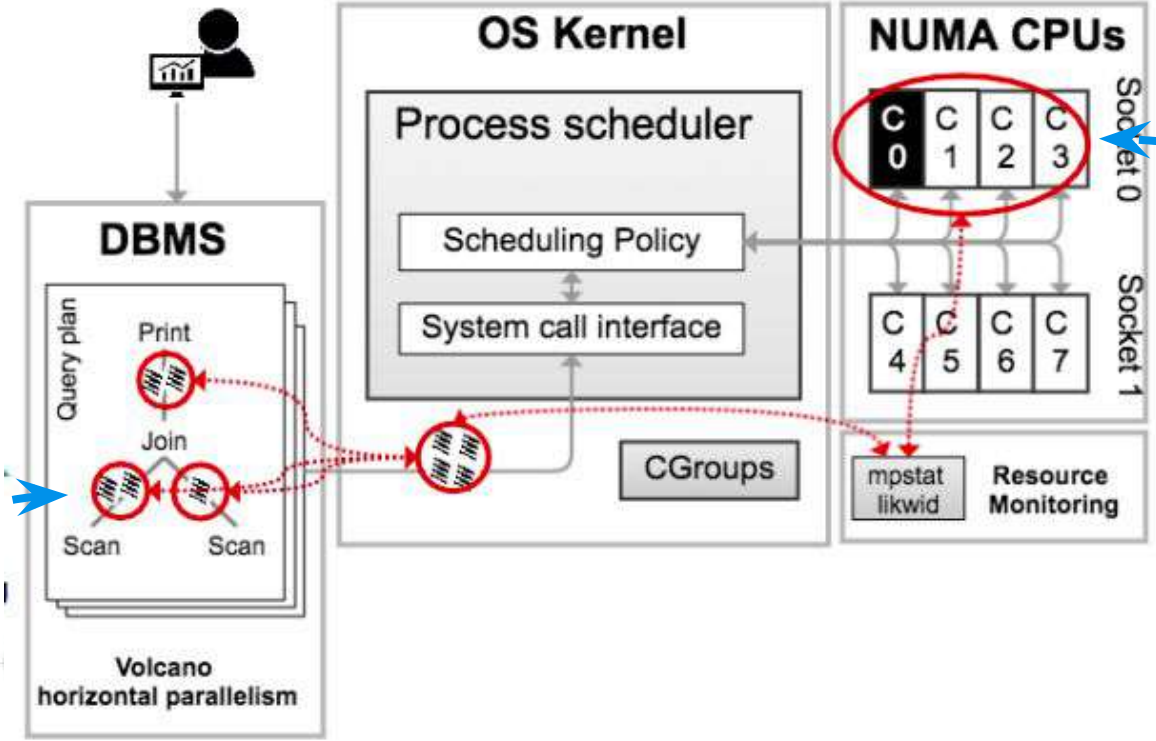
Trabalhos relacionados – SGBD

Trabalho Relacionado	HW	SW	Online	Dinâmico	Threads	Dados	Relacional	n-dimensional	Intrusivo
Dreseler et al. 2017	X		X	X	X		X		
Agrawal et al. 2017	X		X		X		X		SO
Memarzia et al. 2020		X			X	X	X		SO
Ray et al. 2020		X			X	X	X		
Dominico et al. 2022		X	X	X	X		X	X	SGBD

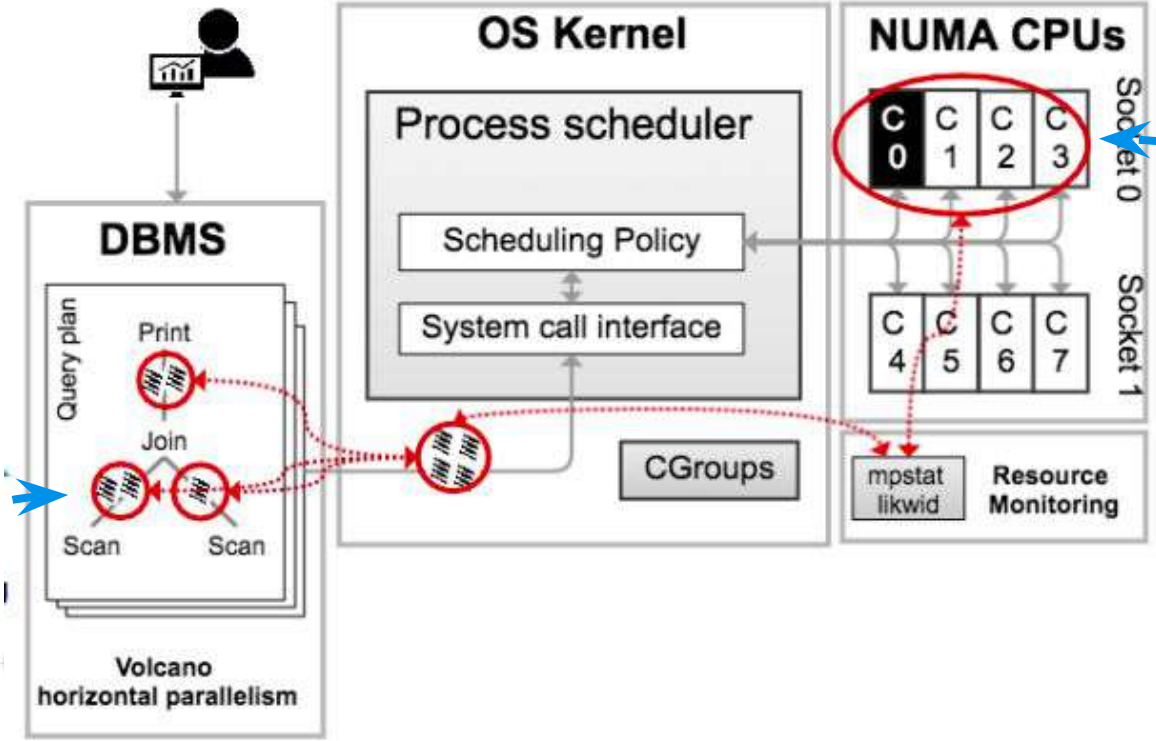
Contribuições

UM MODELO ABSTRATO DE ALOCAÇÃO MULTI-NÚCLEO PARA BANCO DE DADOS RELACIONAL

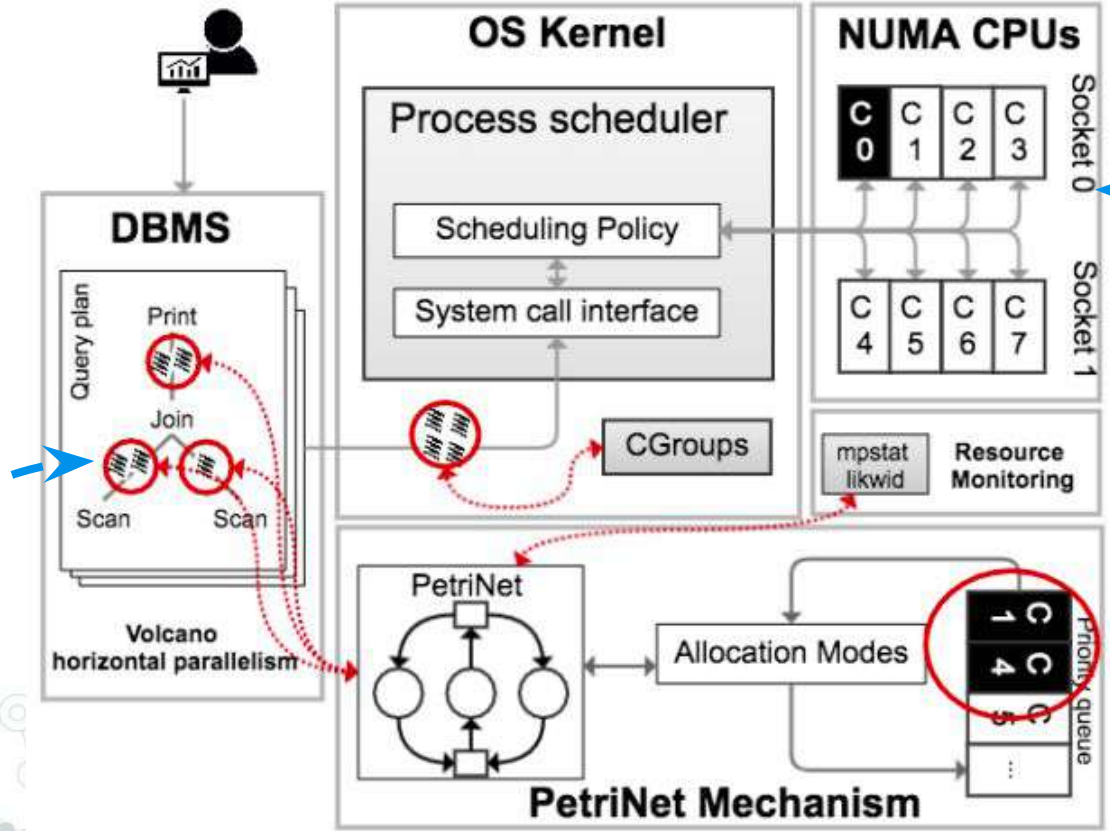
Visão geral do modelo de alocação multi-núcleo



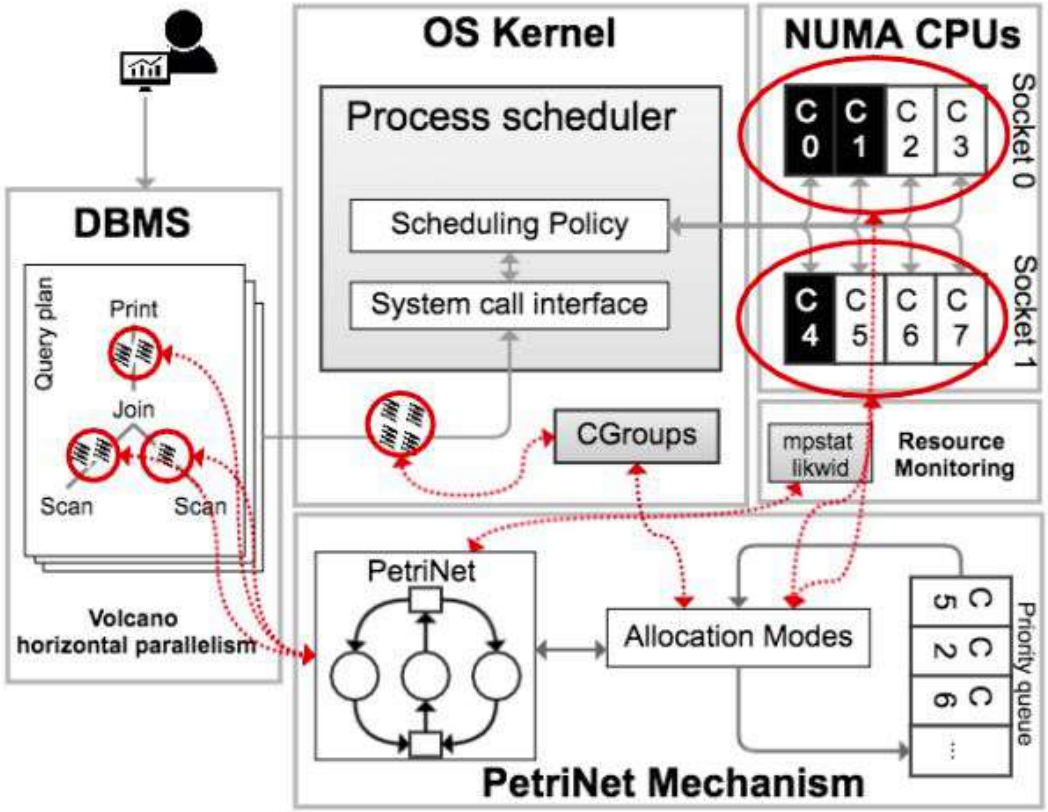
Visão geral do modelo de alocação multi-núcleo



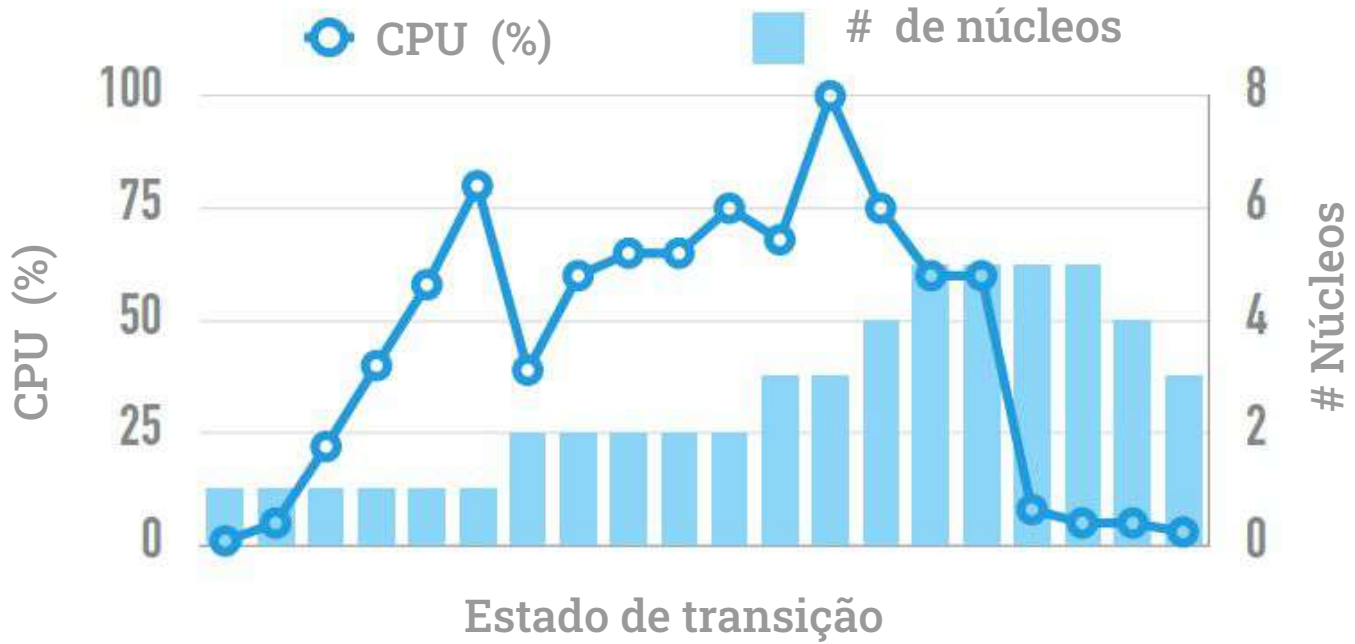
Visão geral do modelo de alocação multi-núcleo



Visão geral do modelo de alocação multi-núcleo

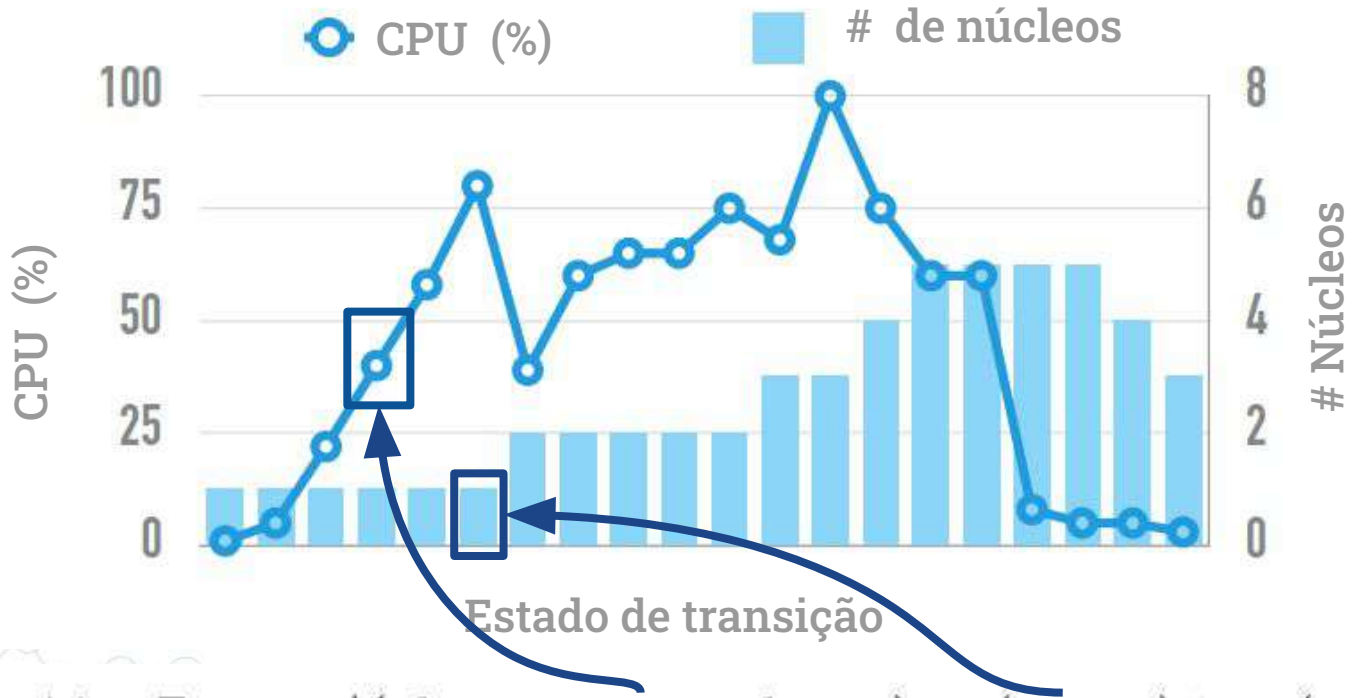


Número ótimo de núcleos



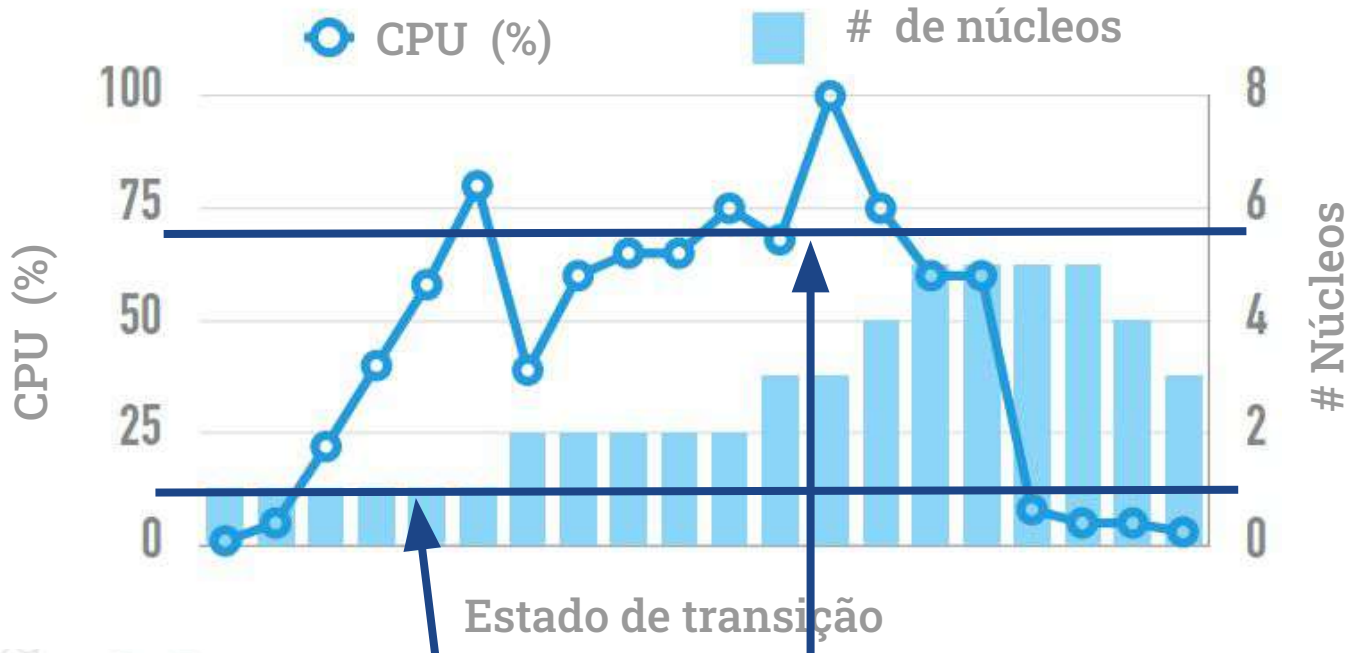
$$\forall\, w\, \exists\, n_{alloc} | (th_{min} < u < th_{max}) \wedge p(n_{alloc}) \geq p(n_{total})$$

Número ótimo de núcleos



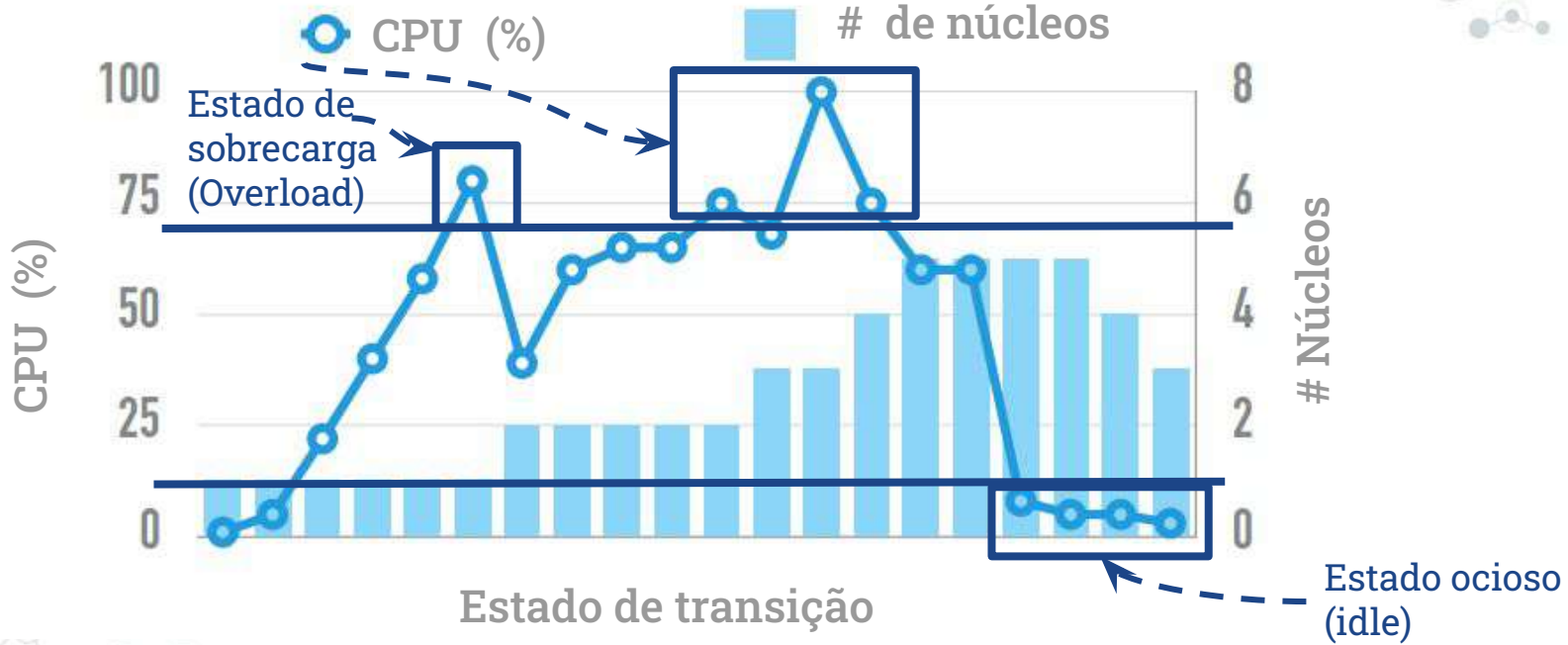
$$\forall\, w\ \exists\, n_{alloc} | (th_{min} < u < th_{max}) \wedge p(n_{alloc}) \geq p(n_{total})$$

Número ótimo de núcleos



$$\forall\, w\ \exists\, n_{alloc} | (th_{min} < u < th_{max}) \wedge p(n_{alloc}) \geq p(n_{total})$$

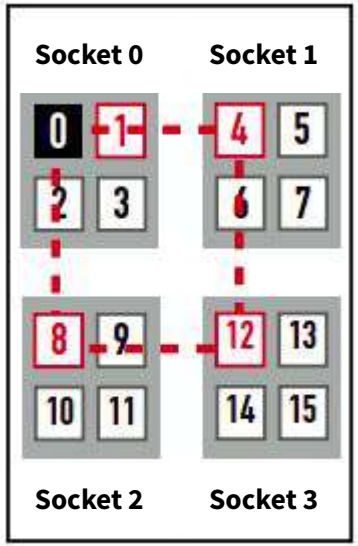
Número ótimo de núcleos



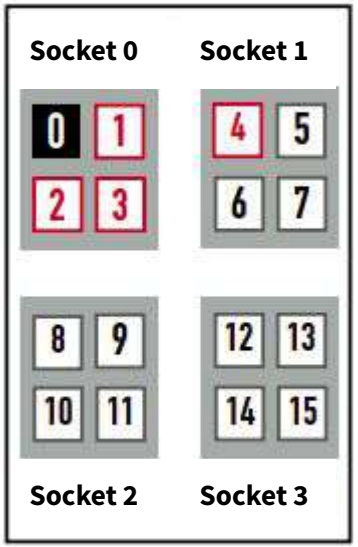
$$\forall w \exists n_{alloc} | (th_{min} < u < th_{max}) \wedge p(n_{alloc}) \geq p(n_{total})$$

Modos de alocação

- **Sparse**



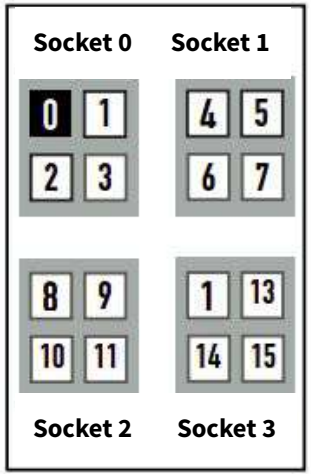
- **Dense**



Alocado

Próximo a ser alocado

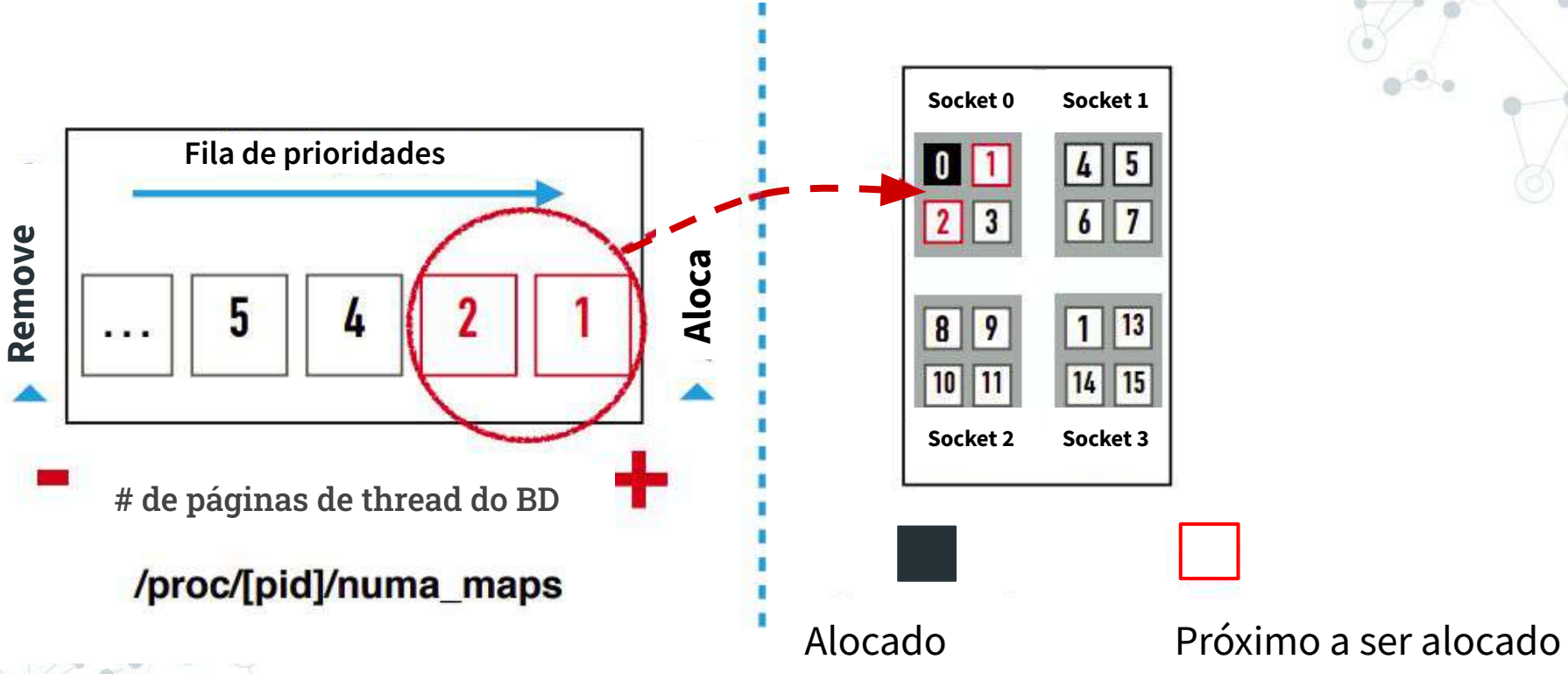
Modo adaptativo



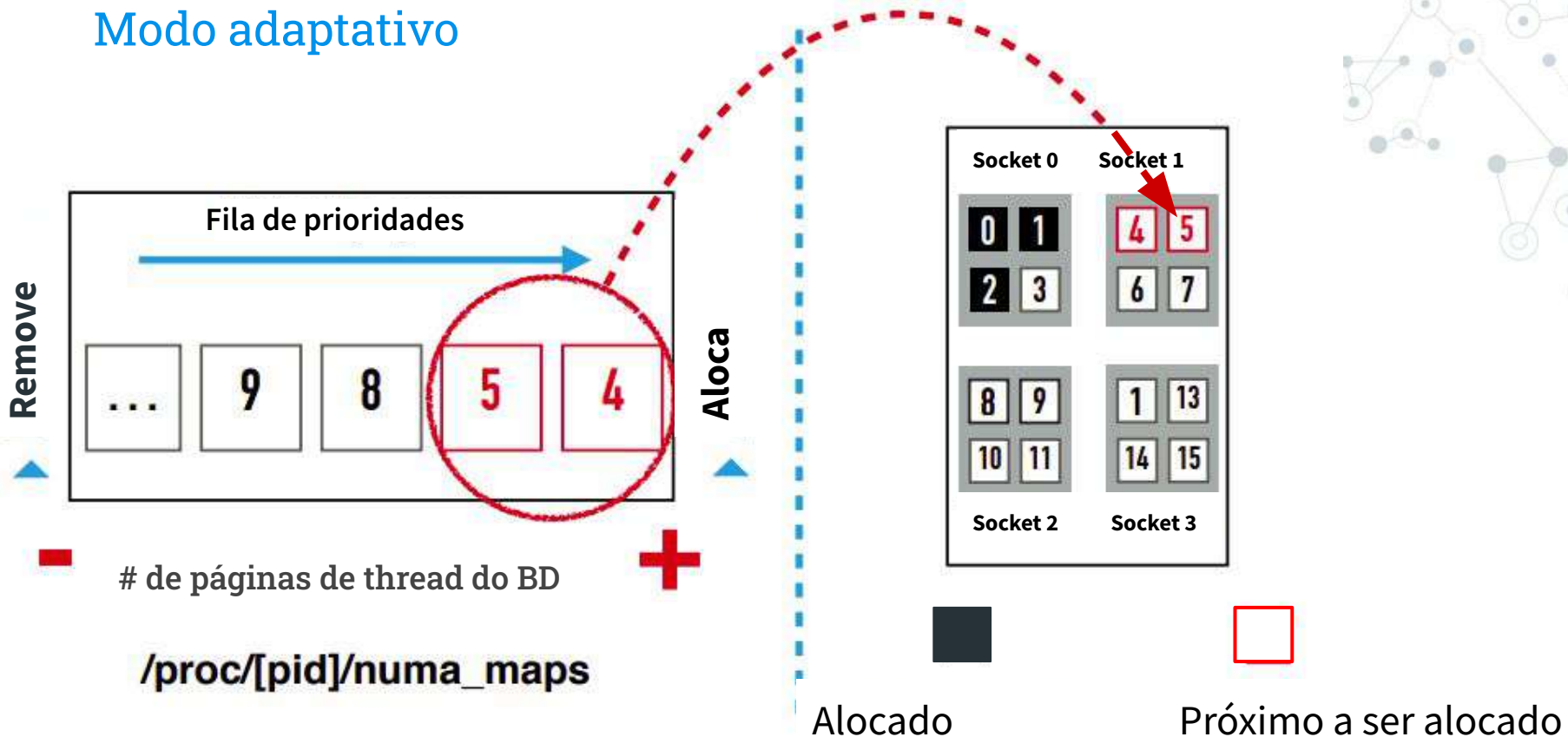
Alocado

Próximo a ser alocado

Modo adaptativo



Modo adaptativo

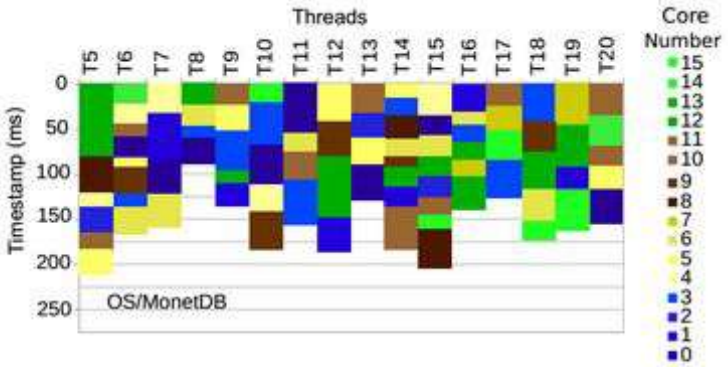


Análise Experimental

Ambiente de Testes

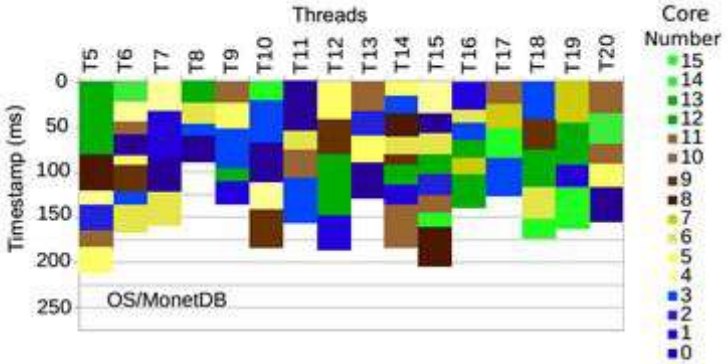
- Base de dados de 1GB do TPC-H.
- 256 usuários
- 1 único usuário
- 4-nós Quad-Core AMD Opteron 8000 Series
  - 64GB RAM/Socket)
  - Largura de banda 41.6 Gb/s
- Debian Linux 8 “Jessie”.
- SGBD:
  - MonetDB (v11.25.5)
  - SQL Server (v2017 Developer RC2)

Impacto da migração de threads

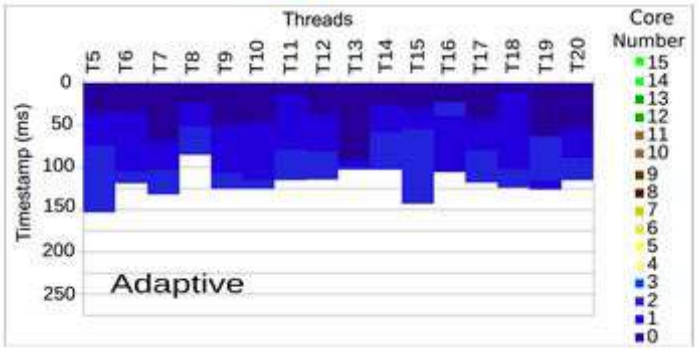


(TPC-H consulta 6, 1 usuário, 1 GB base de dados, MONETDB)

Impacto da migração de threads

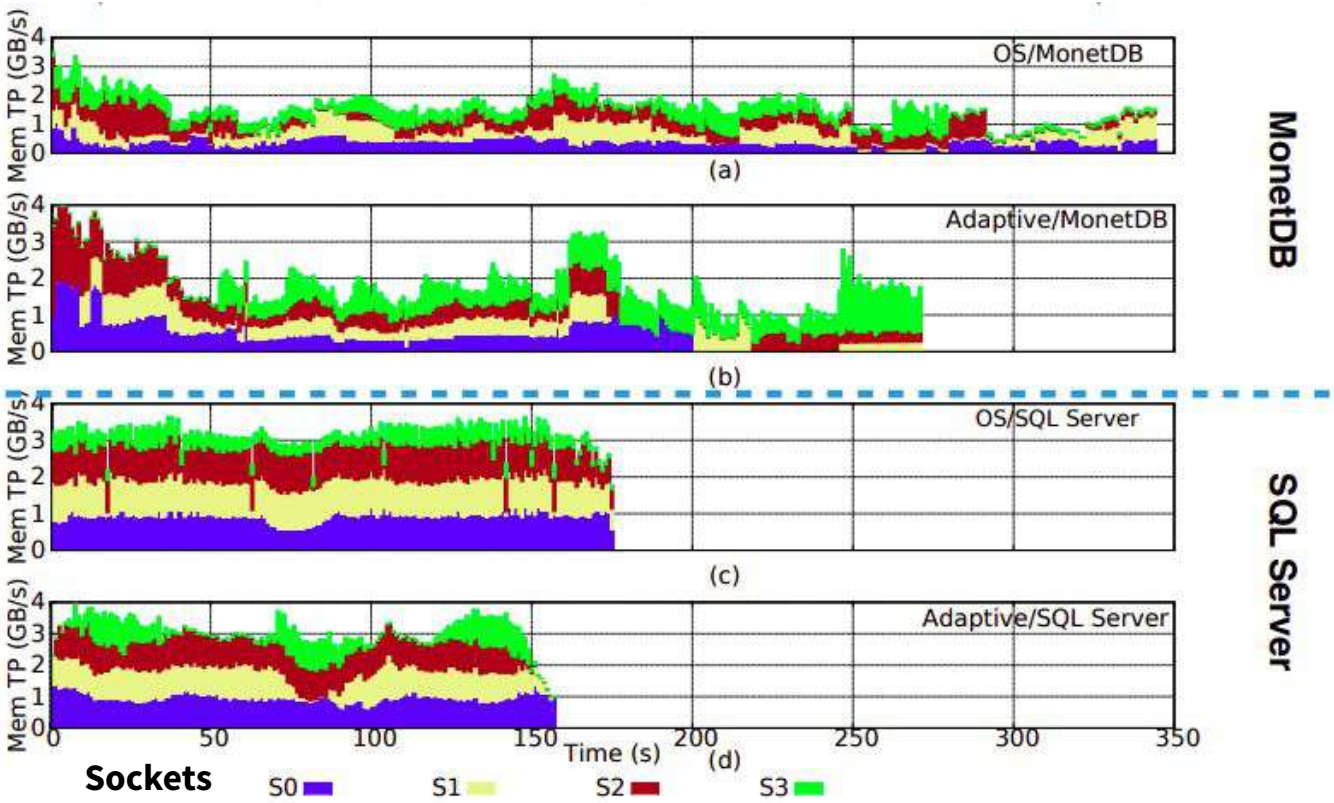


(TPC-H consulta 6, 1 usuário, 1 GB base de dados, MONETDB)



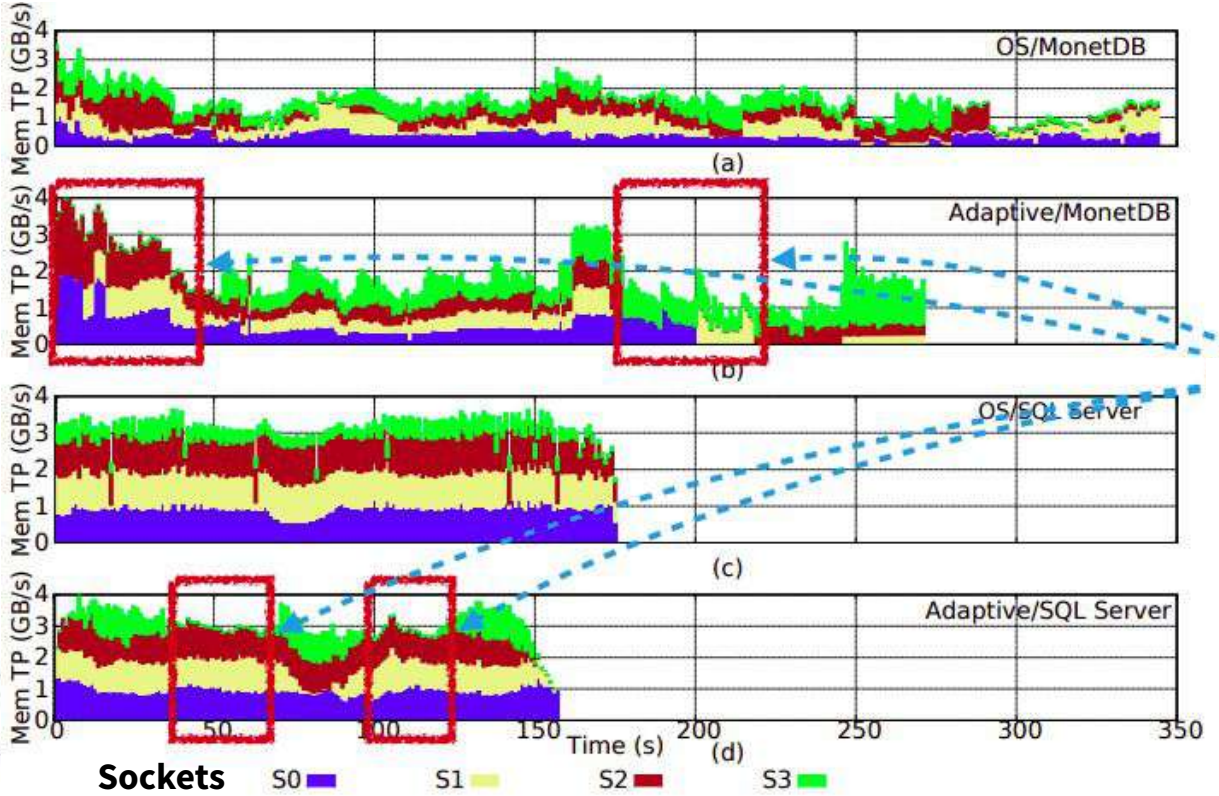
Execução concorrente do TPC-H

256 usuários, base de dados de 1GB, benchmark TPC-H



Execução concorrente do TPC-H

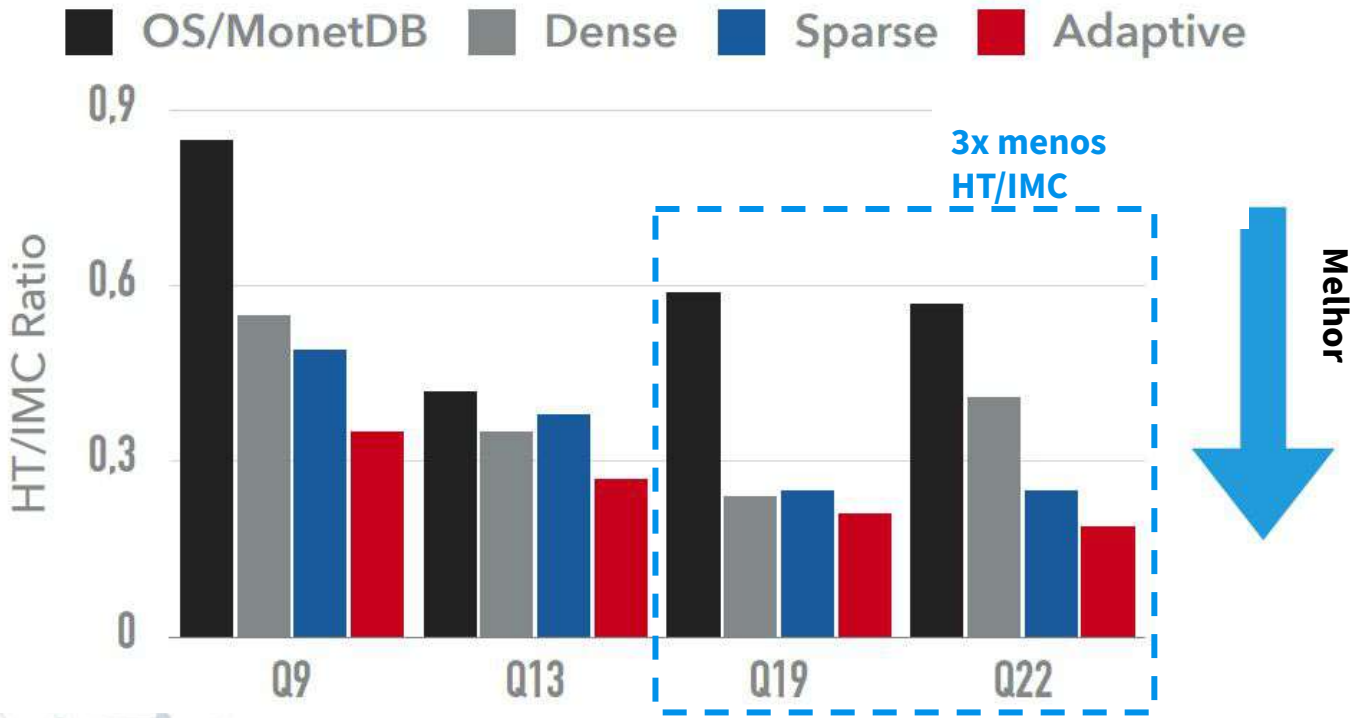
256 usuários, base de dados de 1GB, benchmark TPC-H



Menos núcleos

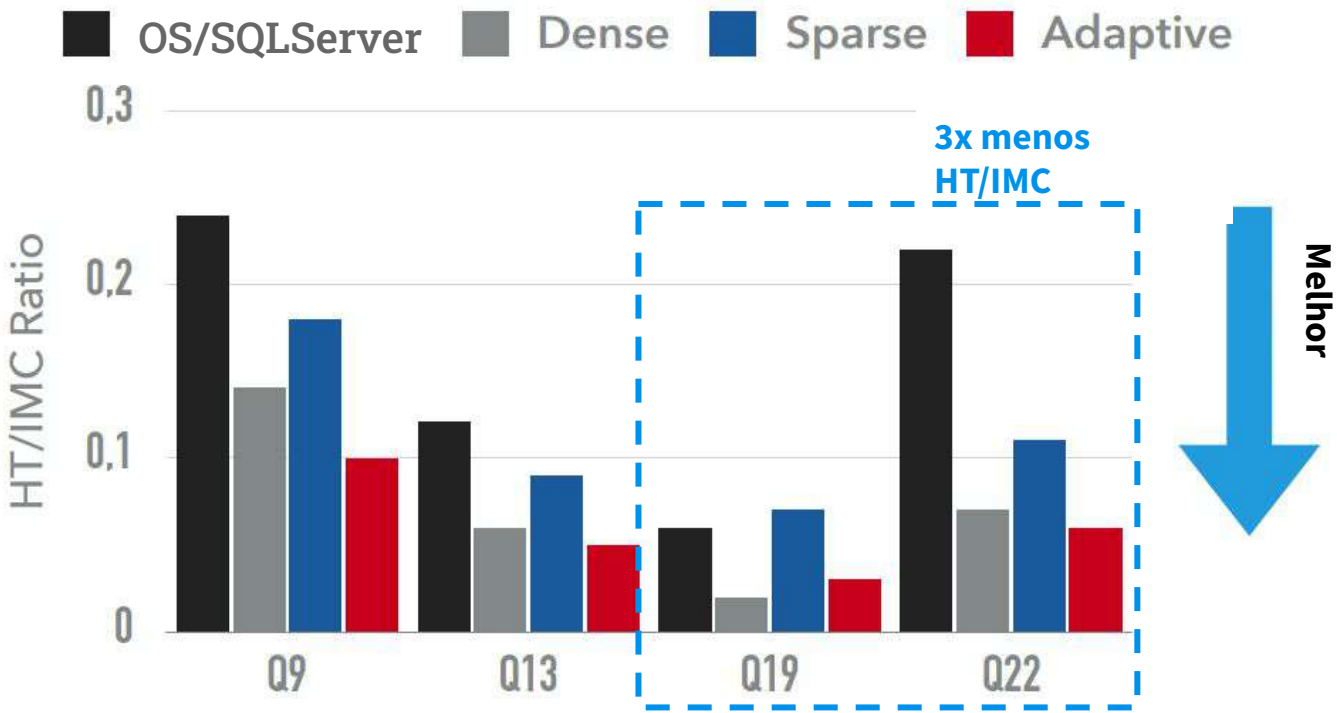
Execução concorrente do TPC-H

MonetDB - 256 usuários, base de dados de 1GB, benchmark TPC-H



Execução concorrente do TPC-H

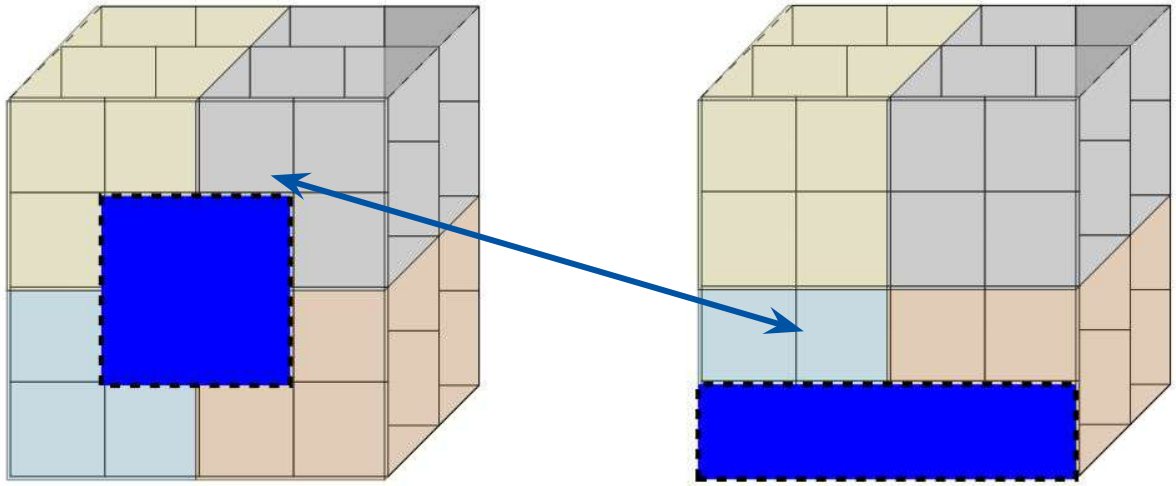
SQLSEVER - 256 usuários, base de dados de 1GB, benchmark TPC-H



ANÁLISE DE DESEMPENHO DE SISTEMAS DE BANCO DE DADOS DE n-dimensional EM ARQUITETURA DE MEMÓRIA NÃO-UNIFORME

Operação subarray banco de dados de n-dimensional

- Seleciona uma parte dos dados
- São repassadas as coordenadas dos dados a serem retornados

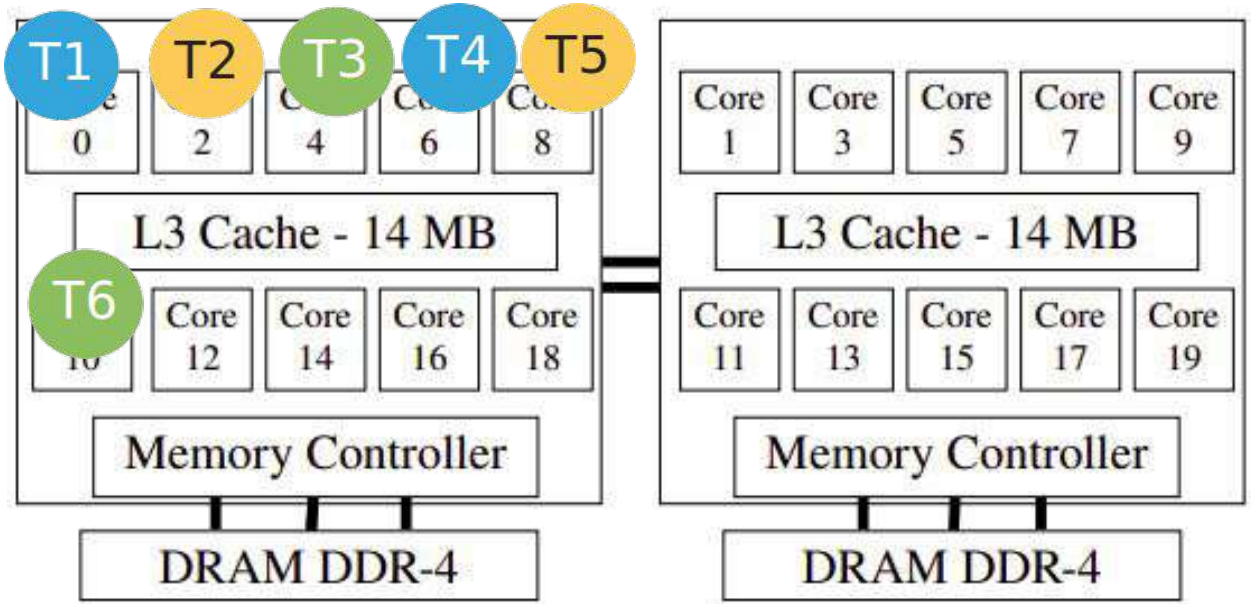


Estratégias de fixação de thread

- Estratégias do estado da arte
- Rede de Petri
- Experimento exaustivo

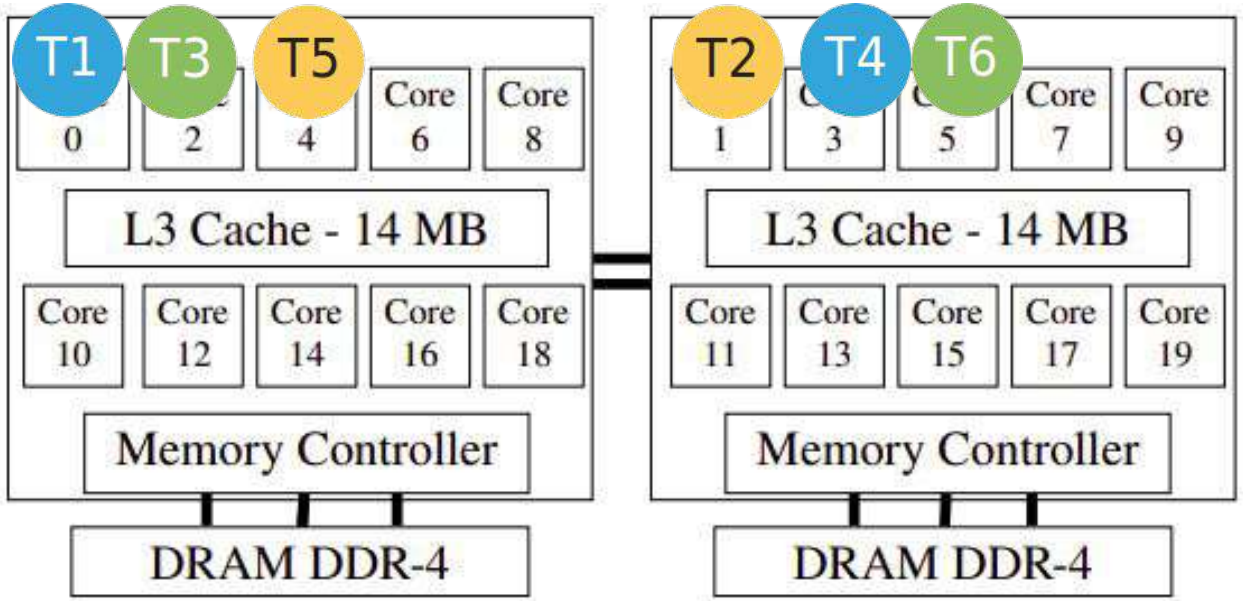
Estratégias de fixação de thread

- Compact - Mesmo nó NUMA



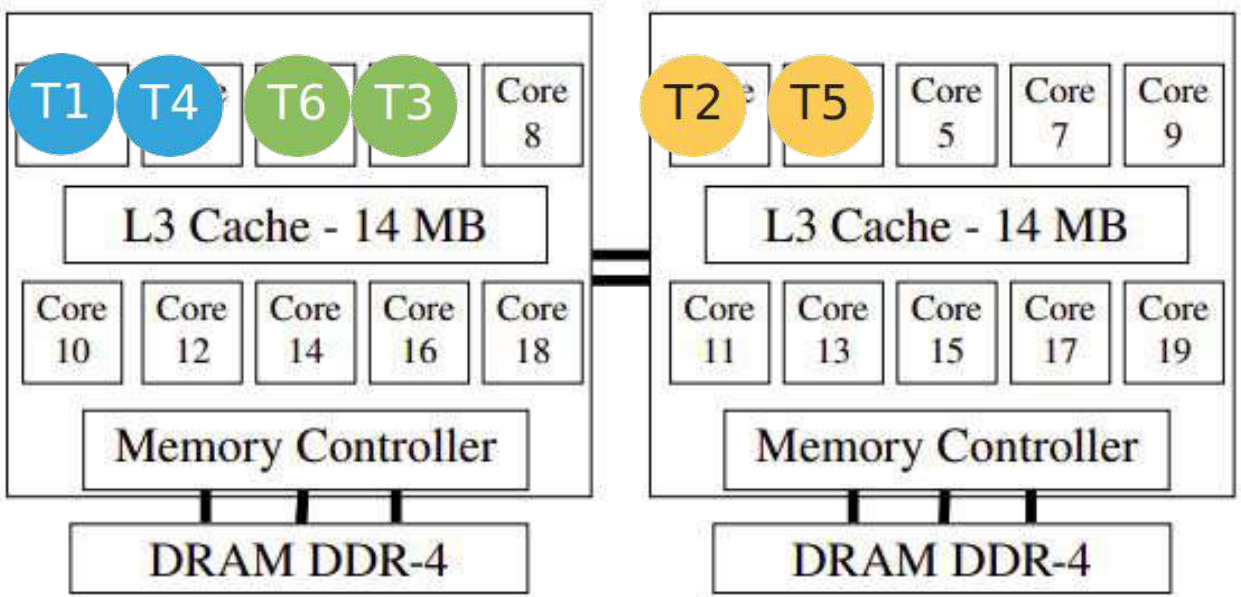
Estratégias de fixação de thread

- Sparse - Uma thread em cada nó NUMA



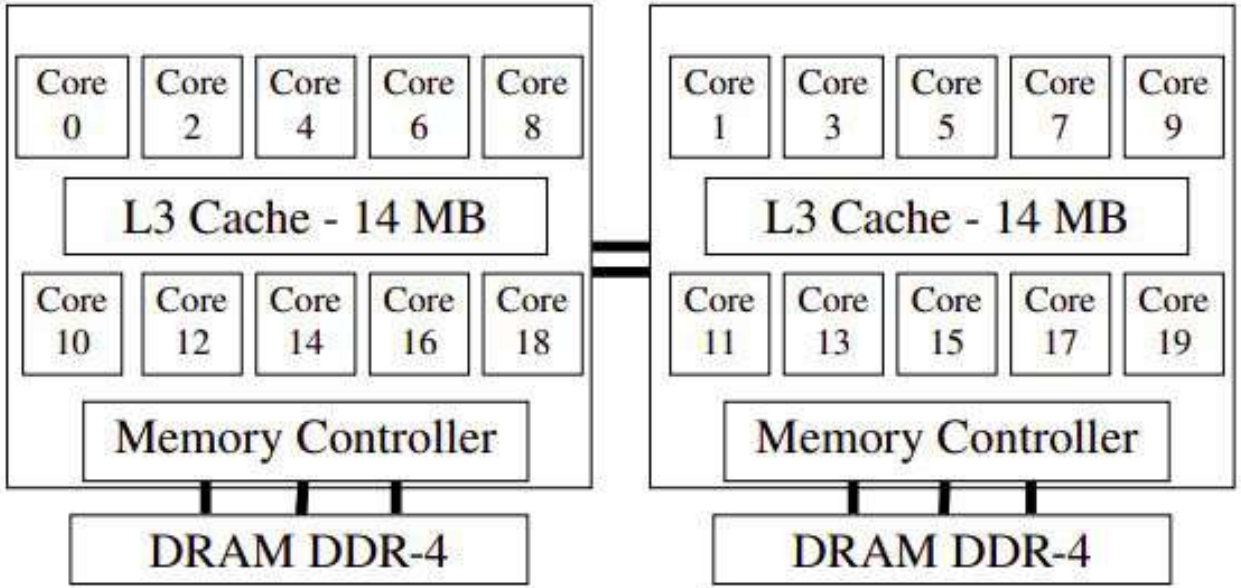
Estratégias de fixação de thread

- Shared - Threads que compartilham dados - mesmo nó



Estratégias de fixação de thread

- Random - selecionado 20 combinações de fixação de thread



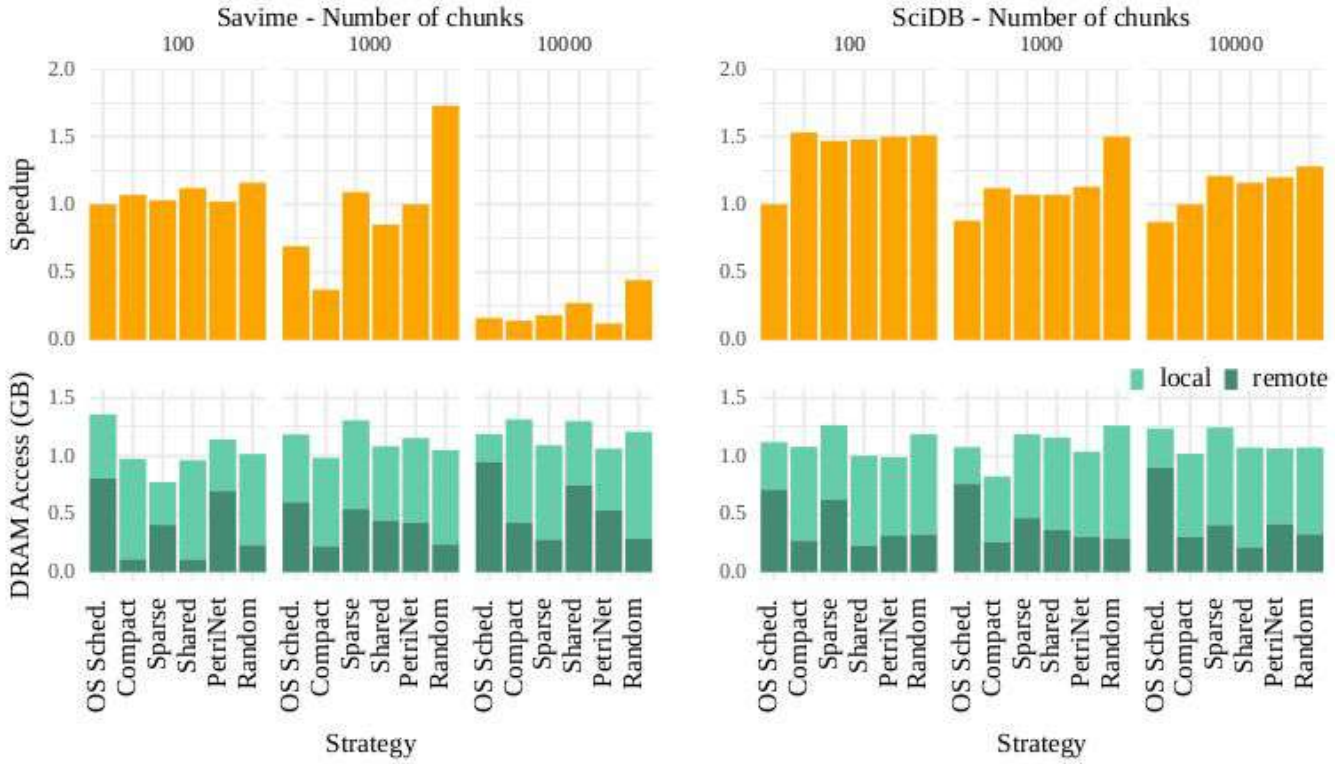
Análise Experimental

Ambiente de Testes

- Subarray: Savime e SciDB
- 2-nós Intel Xeon Silver 4114 (20 núcleos) -- 20 threads
- 2-nós Intel Xeon E5 — 2630 (12 núcleos) -- 12 threads
- Ubuntu 18.04.01 LTS “Bionic Beaver” (Savime)
- Ubuntu 14.04.6 LTS “Trusty Tahr” (SciDB)
- 1GB: quantidades diferentes de chunks
- 50 GB: Seletividade

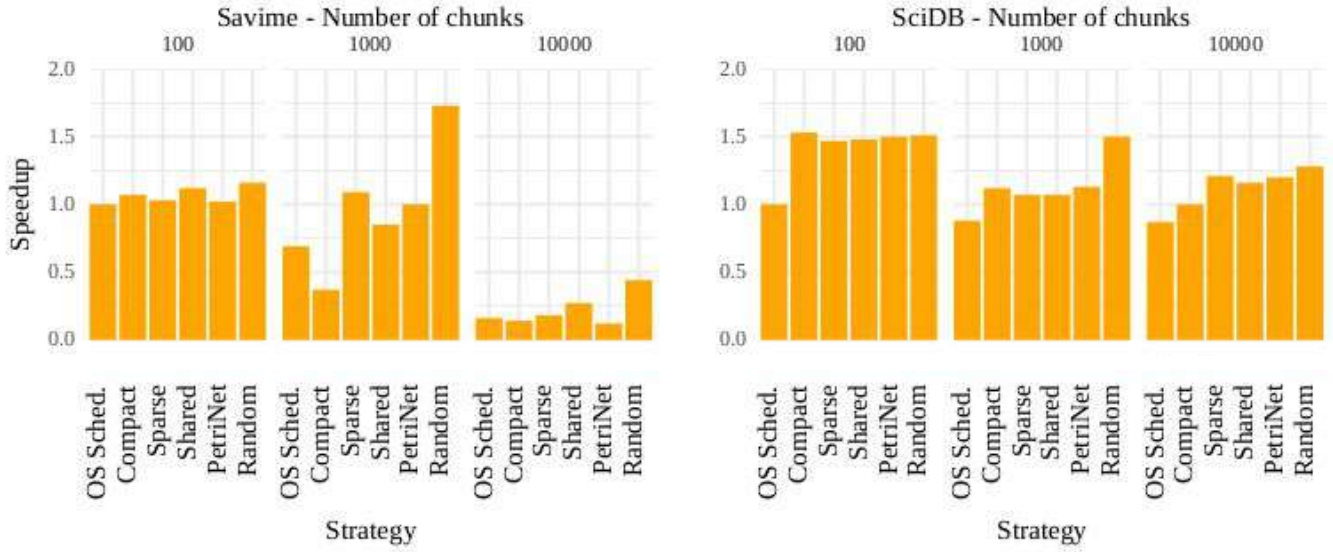
Comparação de performance operador subarray em uma base de dados de 1 GB

Melhor



Comparação de performance operador subarray em uma base de dados de 1 GB

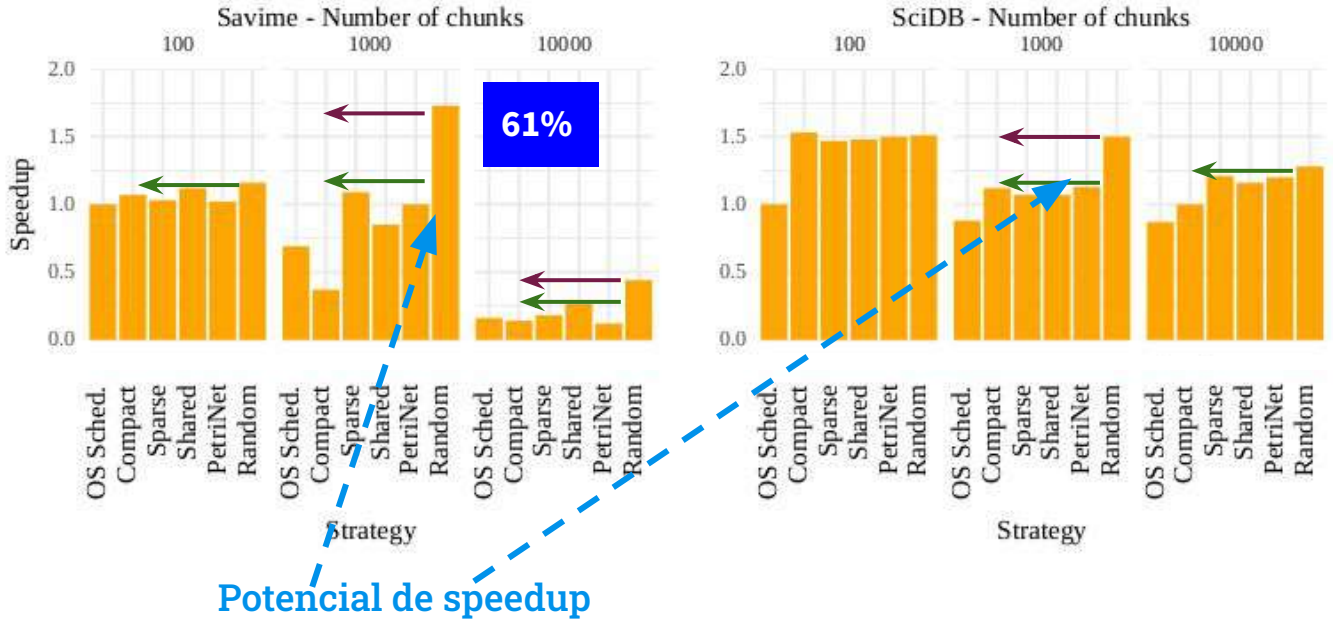
Melhor



- Speedup médio de 1,52x
- Dados normalizados por 100 chunks

Comparação de performance operador subarray em uma base de dados de 1 GB

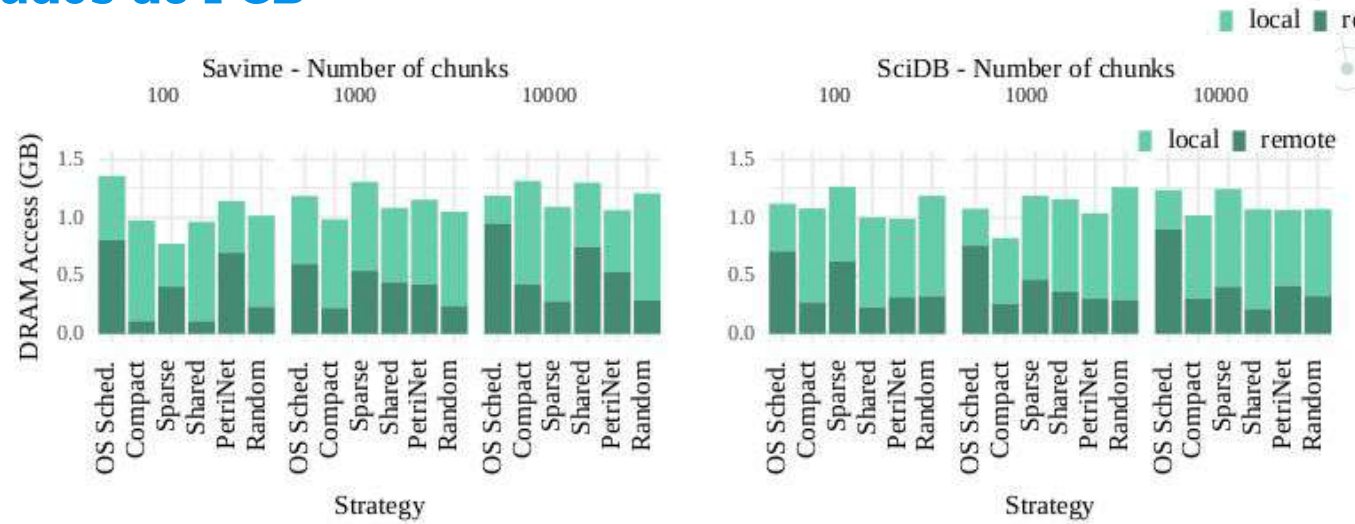
Melhor



- Média de 52% de speedup
- Dados normalizados por 100 chunks

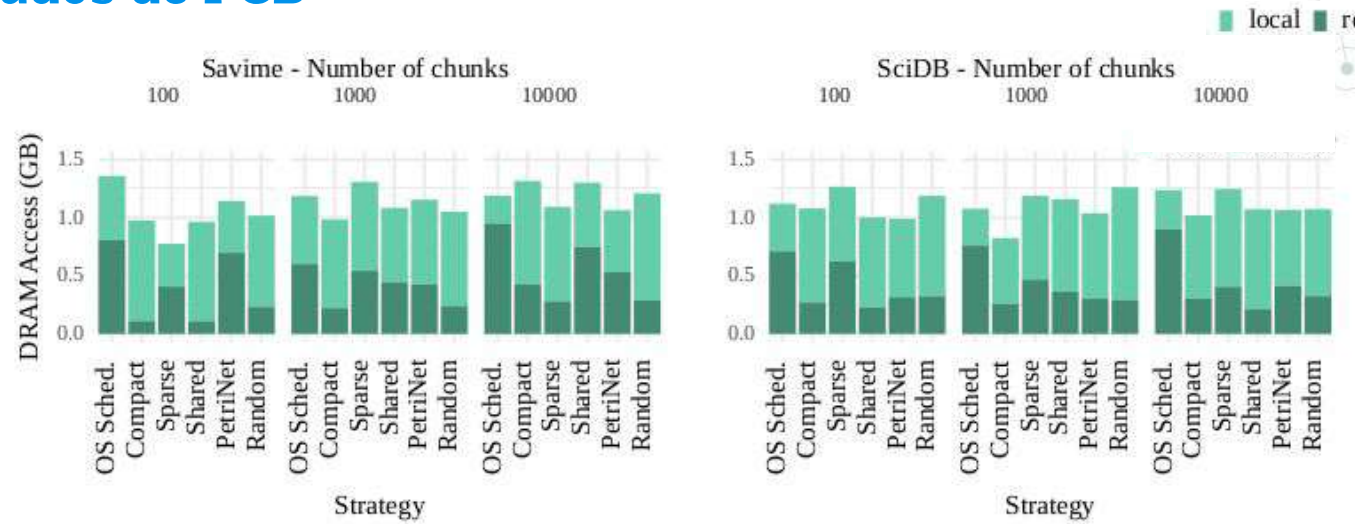
Comparação de performance operador subarray em uma base de dados de 1 GB

Melhor



Comparação de performance operador subarray em uma base de dados de 1 GB

Melhor



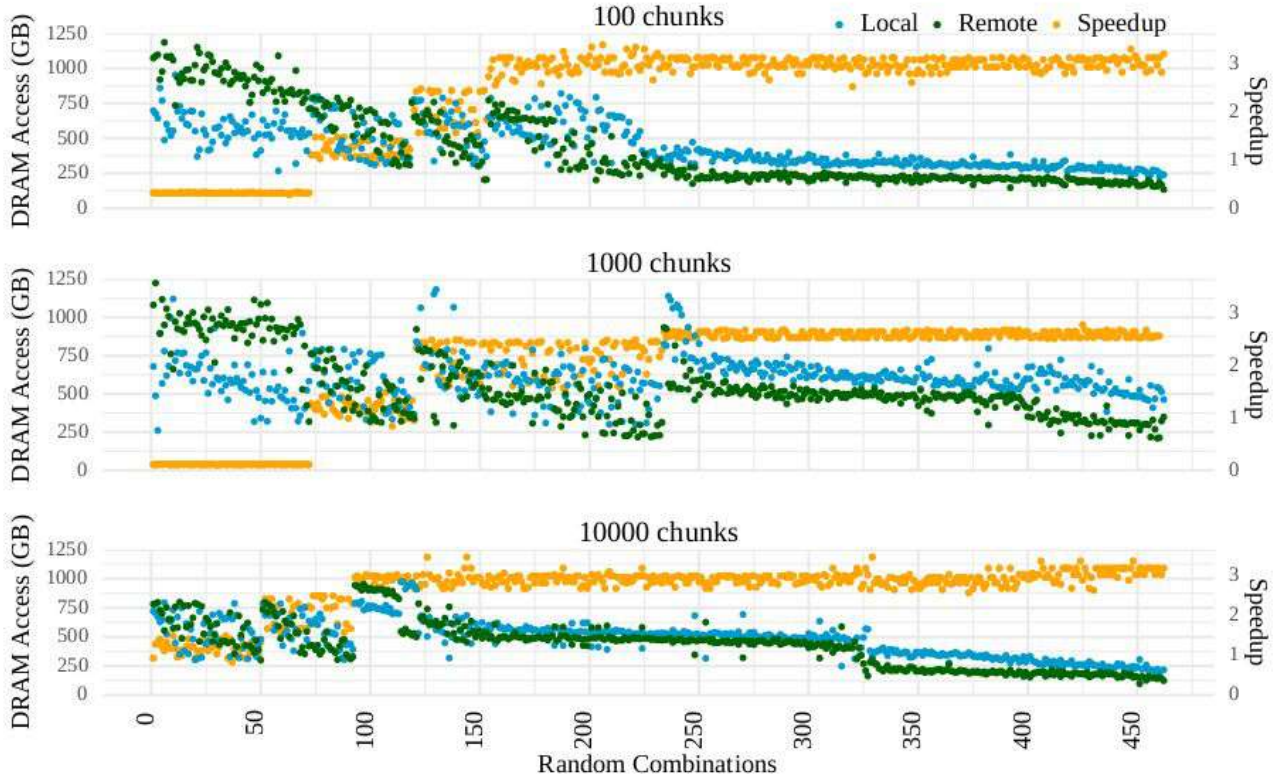
- SAVIME: 4.9x menos acessos remotos
- SciDB: 3.5x menos acessos remotos

Sumarização de resultados de performance operador subarray - 50 GB

- **SAVIME**
  - Random baixos acessos remotos menos **5x** acessos remotos
  - **3,68%** de potencial de speedup
  - Alta seletividade desempenho melhor
  - 68% redução no consumo de energia
- **SciDB**
  - Random menos **4,1x** acessos remotos
  - Mapeamentos com resultados diferentes
  - Sparse tem melhor desempenho
  - 16% redução no consumo de energia

Avaliação exaustiva das possibilidades de fixação de thread – Speedup e Acesso à memória

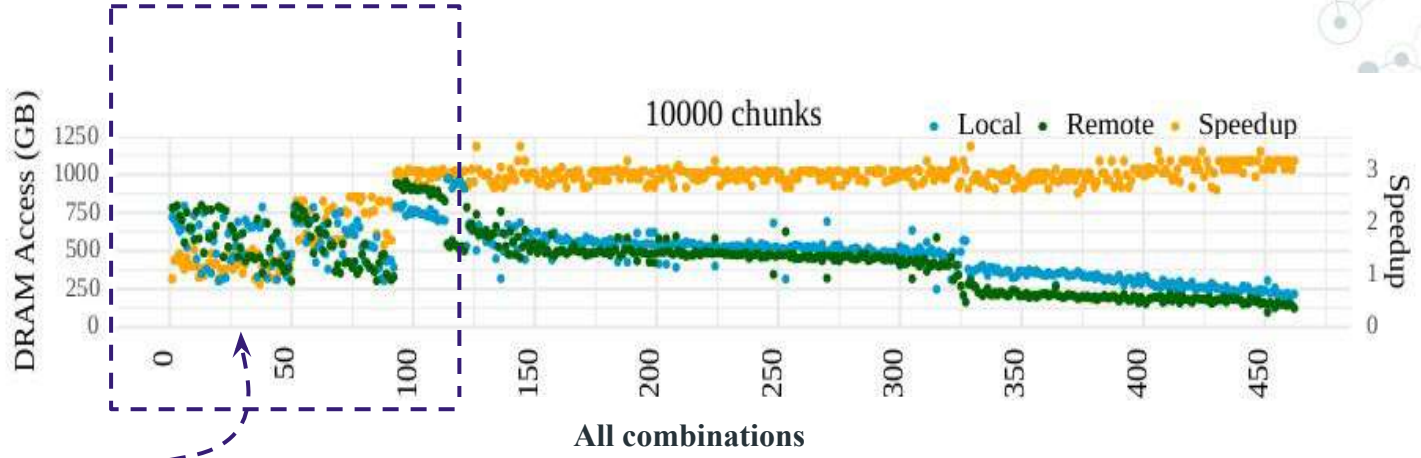
Melhor



Melhor

Avaliação exaustiva das possibilidades de fixação de thread — Speedup e Acesso à memória

Melhor

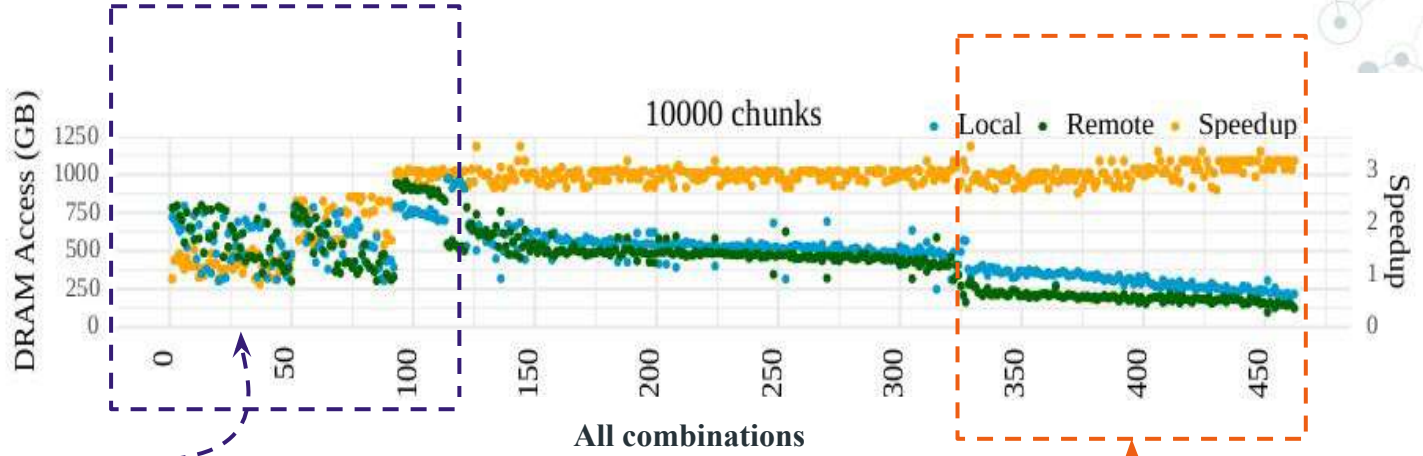


Melhor

- 7% das 462 possibilidades perdem desempenho
- Apresenta 2,5 × mais acessos à memória.

Avaliação exaustiva das possibilidades de fixação de thread — Speedup e Acesso à memória

Melhor



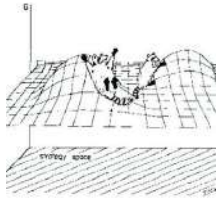
Melhor

- 7% das 462 possibilidades perdem desempenho
- Apresenta 2,5 × mais acessos à memória.
- Menos acessos remotos a memória = speedup maior

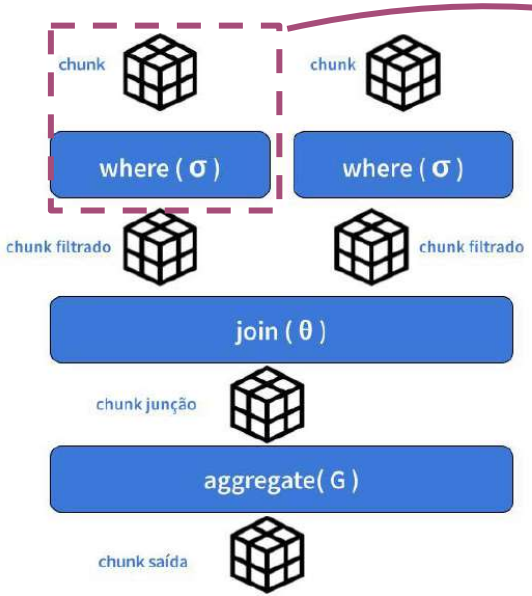
Pergunta em aberto

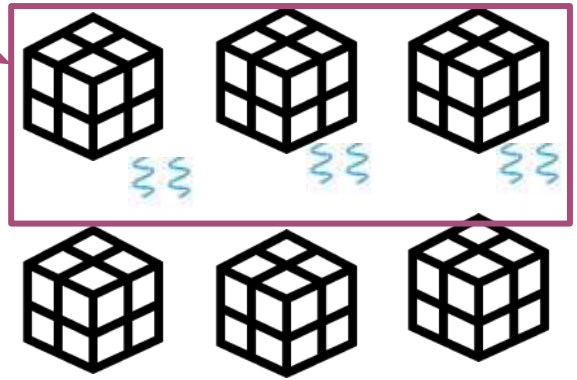
Como determinar o melhor posicionamento das threads de um banco de dados multidimensional na arquitetura NUMA?

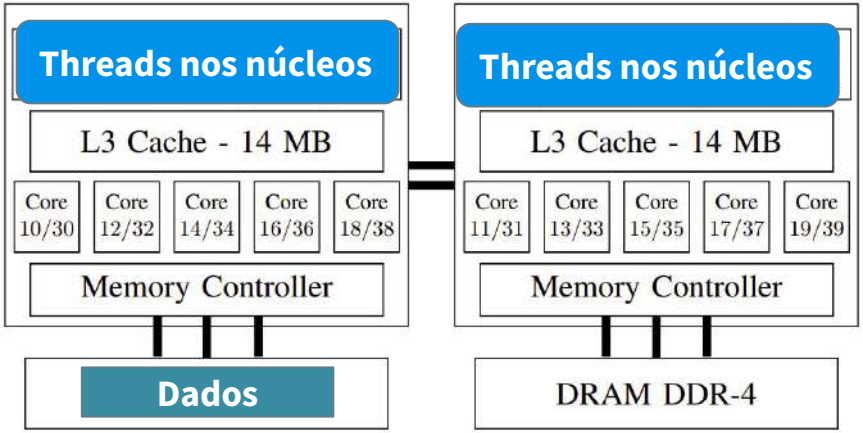
NASARRAY: UM JOGO NÃO-COOPERATIVO PARA ALOCAÇÃO DE THREADS DE BANCO DE DADOS DE N-DIMENSIONAL



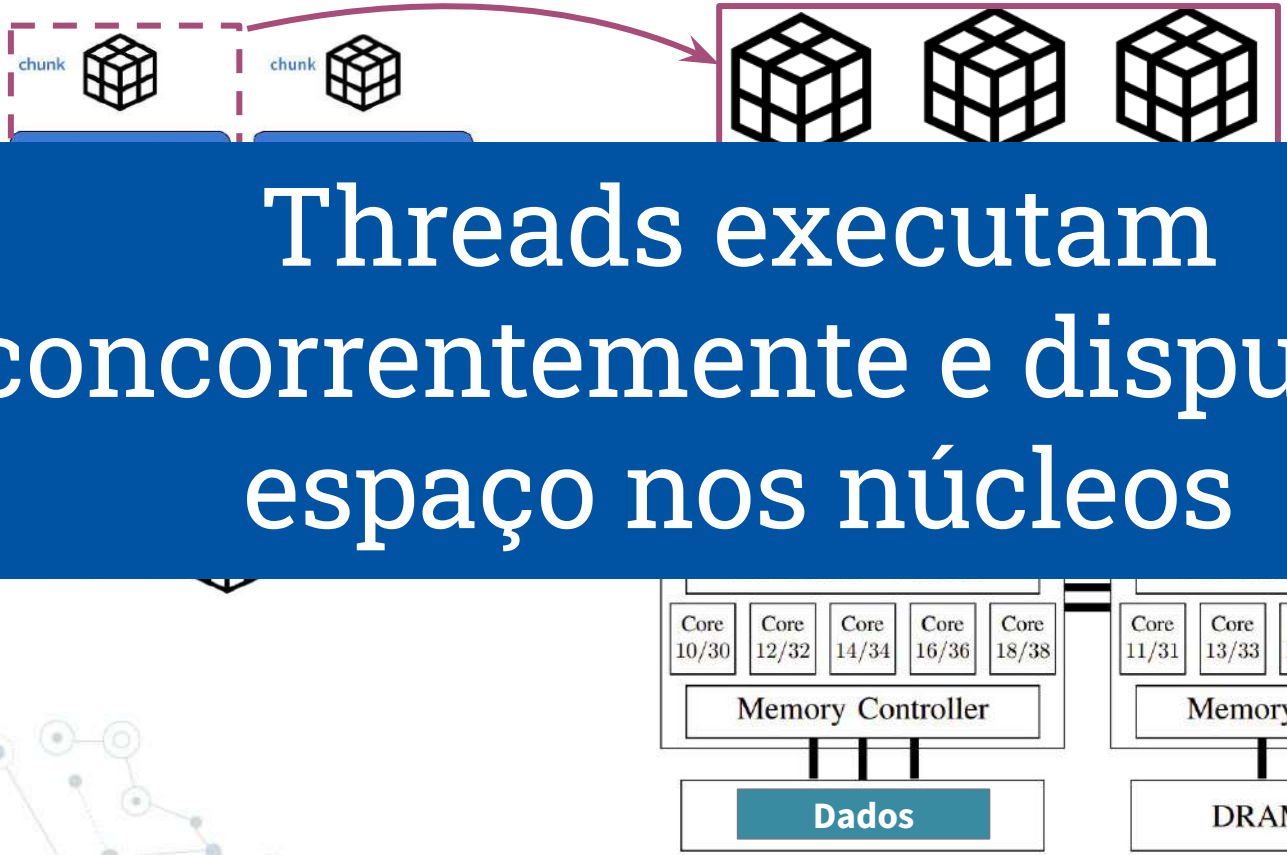
Posicionamento de threads pelo SO







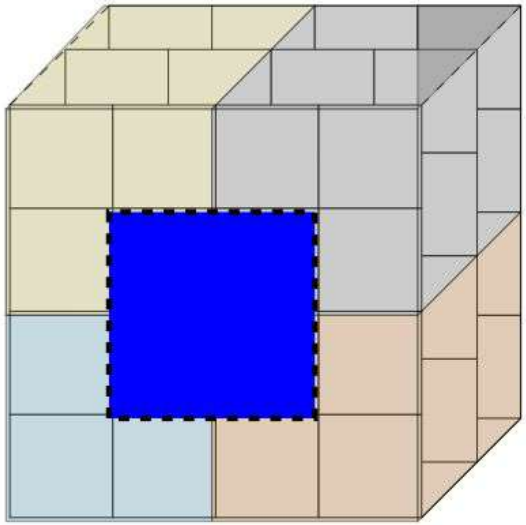
Posicionamento de threads pelo SO

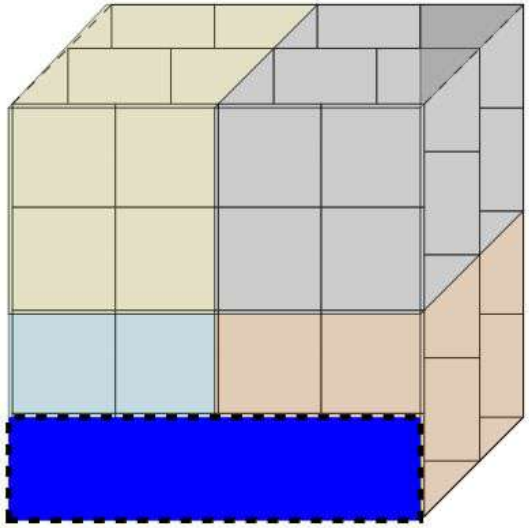


Threads executam concorrentemente e disputam espaço nos núcleos

Operação subarray banco de dados de n-dimensional

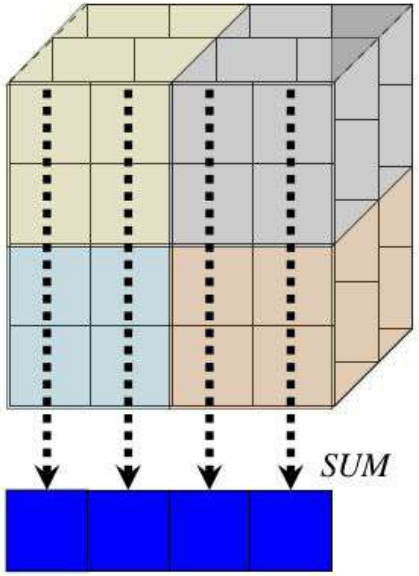
- baixo reuso de dados
- acesso à memória coalescente (BC)





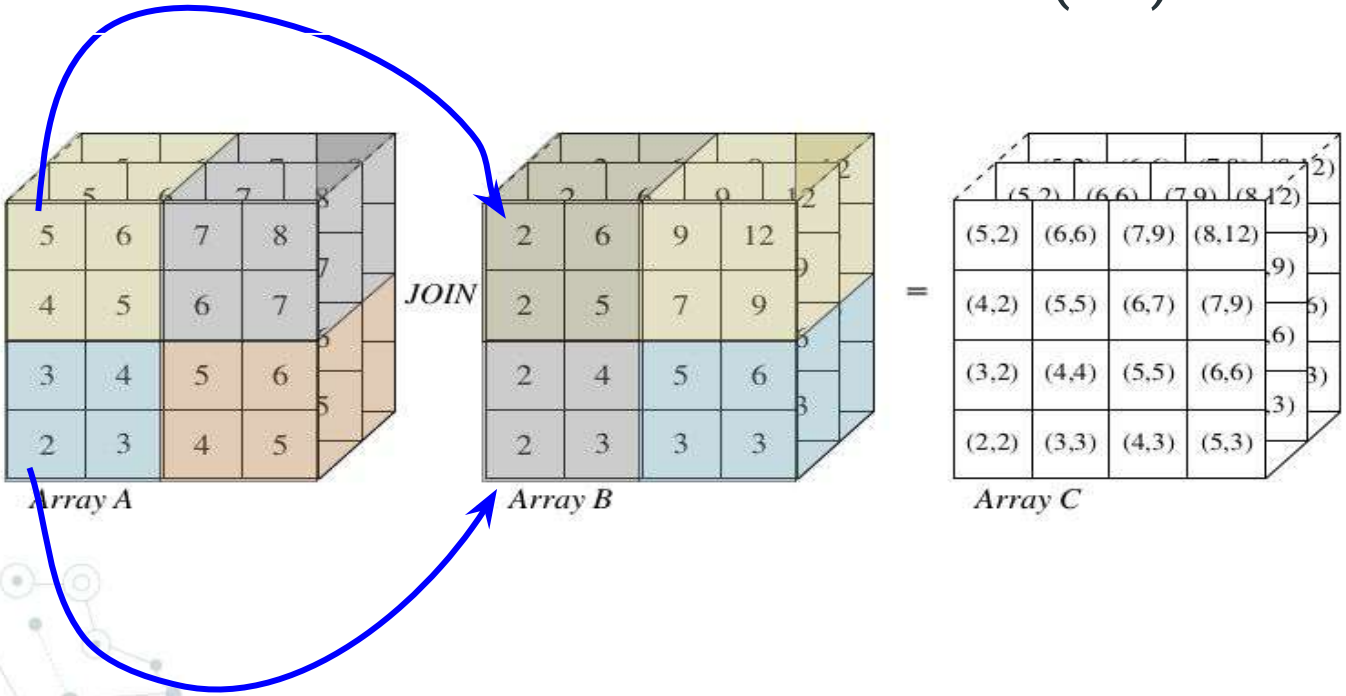
Operações de banco de dados de n-dimensional -- Agregação

- alto reuso de dados (AR).
- Acesso randômico a memória.



Operações de banco de dados de n-dimensional -- Junção

- alto reuso de dados
- com acesso à memória coalescente (AC).



Threads como agente tomador de decisão -- Definição

- **Definição 1.** *Jogador*: um jogador é uma thread do banco de dados.

Threads como agente tomador de decisão -- Definição

- **Definição 1.** *Jogador*: um jogador é uma thread do banco de dados.
- **Definição 2.** *Ação ou movimento:* qual nó NUMA que a thread pode se posicionar em um dado momento do jogo.

Threads como agente tomador de decisão -- Definição

- **Definição 1.** *Jogador*: um jogador é uma thread do banco de dados.
- **Definição 2.** *Ação ou movimento:* qual nó NUMA que a thread pode se posicionar em um dado momento do jogo.
- **Definição 3.** *Estratégias:* Quais ações disponíveis para cada jogador?

Threads como agente tomador de decisão -- Definição

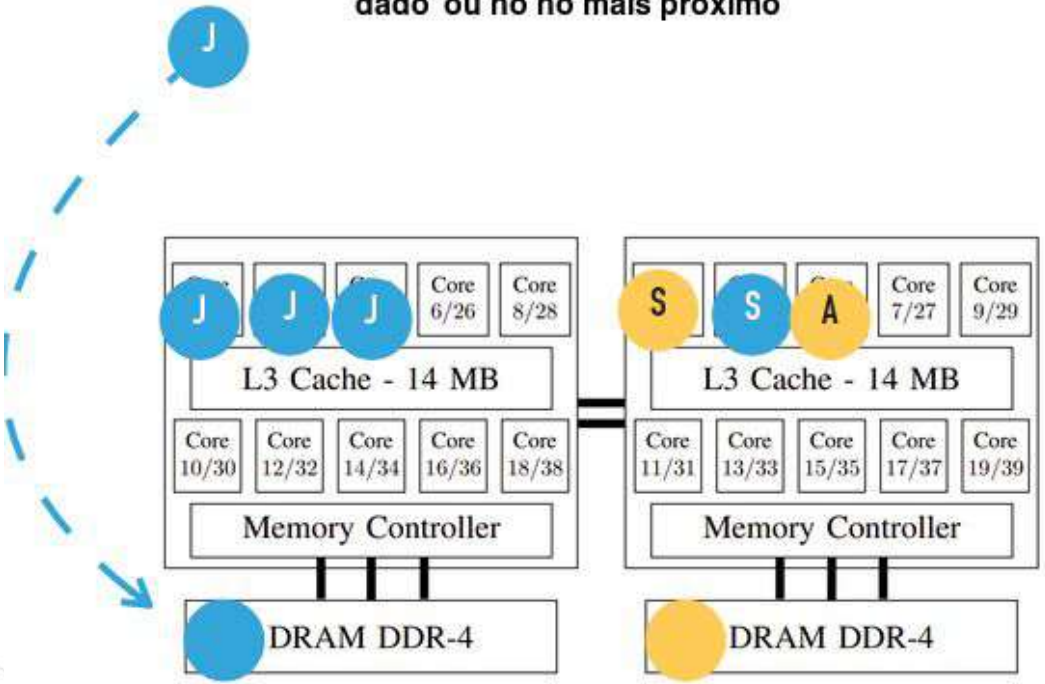
- **Definição 1.** *Jogador*: um jogador é uma thread do banco de dados.
- **Definição 2.** *Ação ou movimento:* qual nó NUMA que a thread pode se posicionar em um dado momento do jogo.
- **Definição 3.** *Estratégias:* Quais ações disponíveis para cada jogador?
- **Definição 4**. *Estratégia Pura:* O jogador faz uma escolha específica

Threads como agente tomador de decisão -- Definição

- **Definição 1.** *Jogador*: um jogador é uma thread do banco de dados.
- **Definição 2.** *Ação ou movimento:* qual nó NUMA que a thread pode se posicionar em um dado momento do jogo.
- **Definição 3.** *Estratégias:* Quais ações disponíveis para cada jogador?
- **Definição 4**. *Estratégia Pura:* O jogador faz uma escolha específica
- **Definição 5.** *Equilíbrio de Nash:* “Estou fazendo o melhor que posso, dado o que você está fazendo” [Nash, 1951].

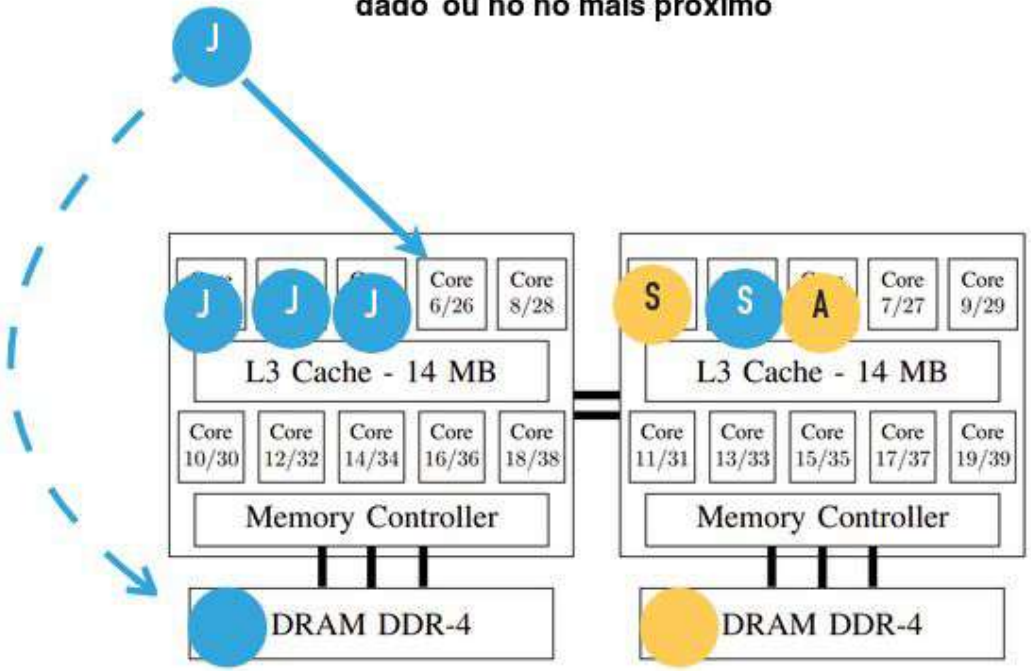
Threads como agente tomador de decisão

$\forall\, t_i \mid f_{t1}(x) = 0$, t_i se posiciona no nó onde está seu dado ou no nó mais próximo

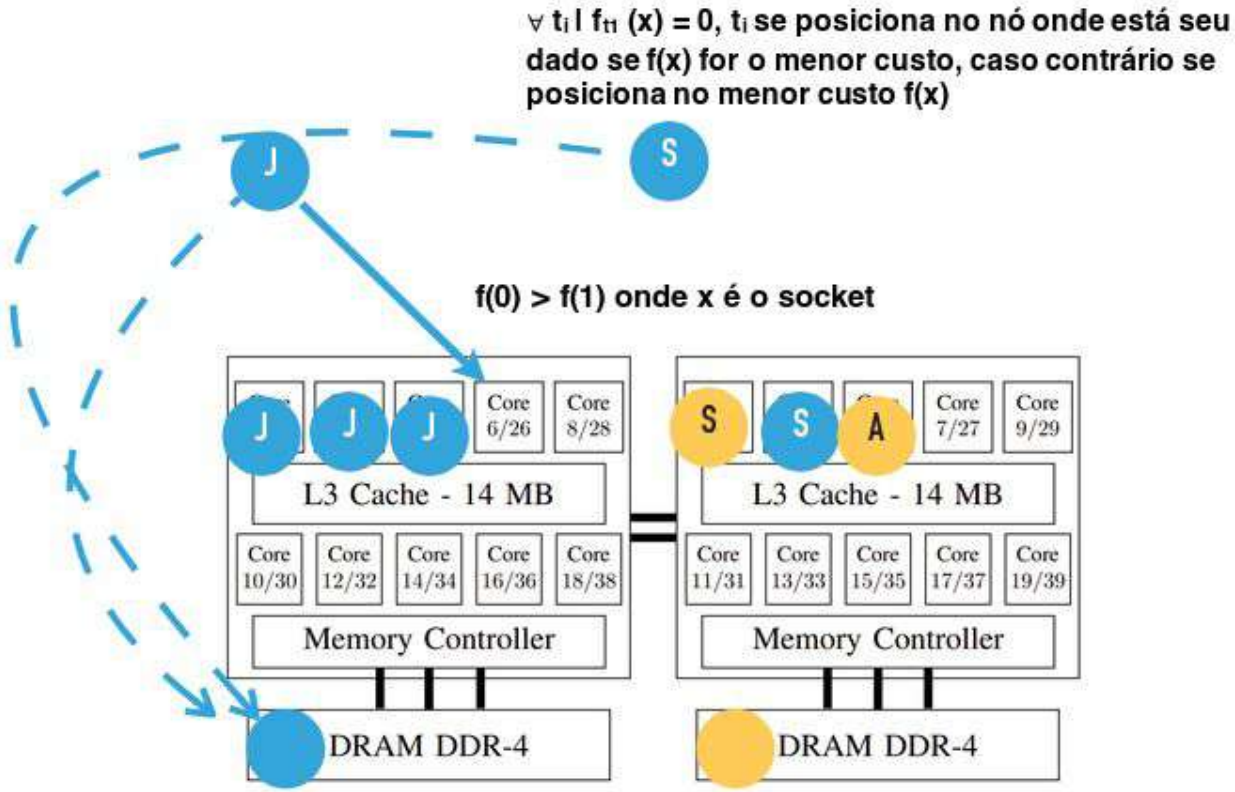


Threads como agente tomador de decisão

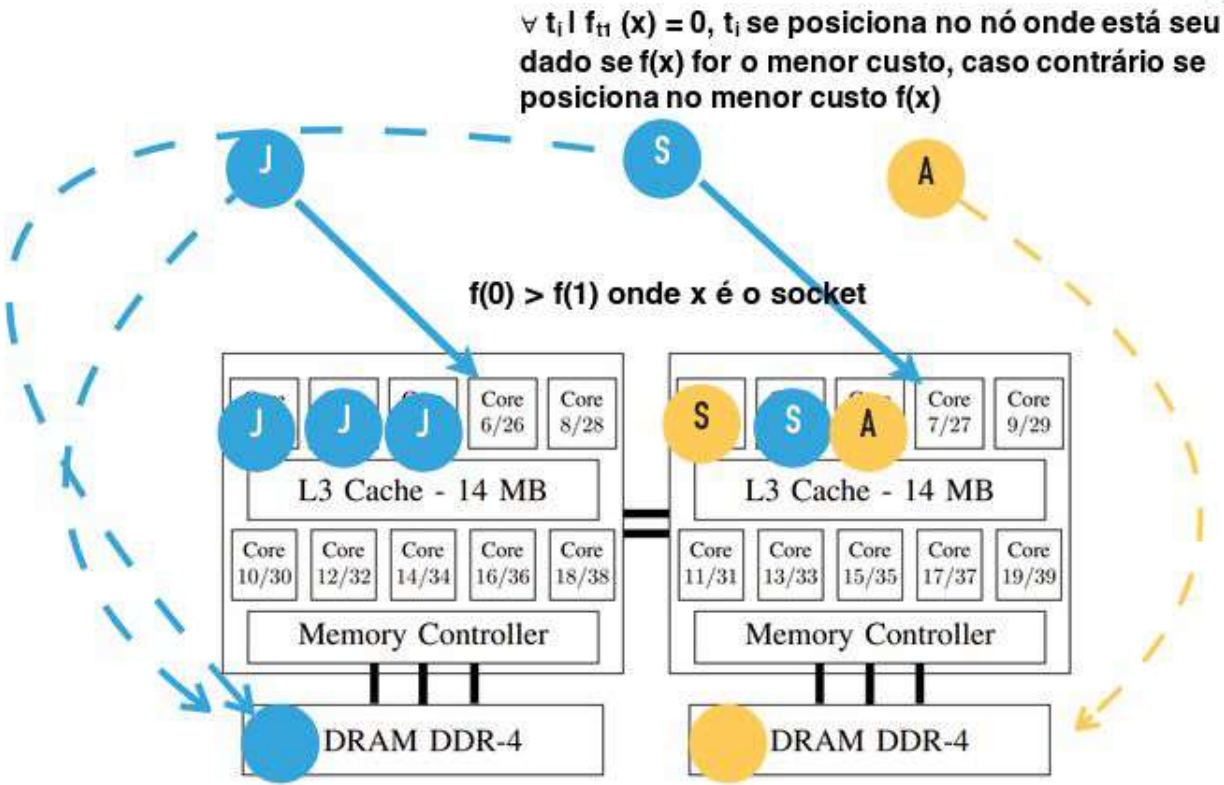
$\forall\, t_i \mid f_{t1}(x) = 0$, t_i se posiciona no nó onde está seu dado ou no nó mais próximo



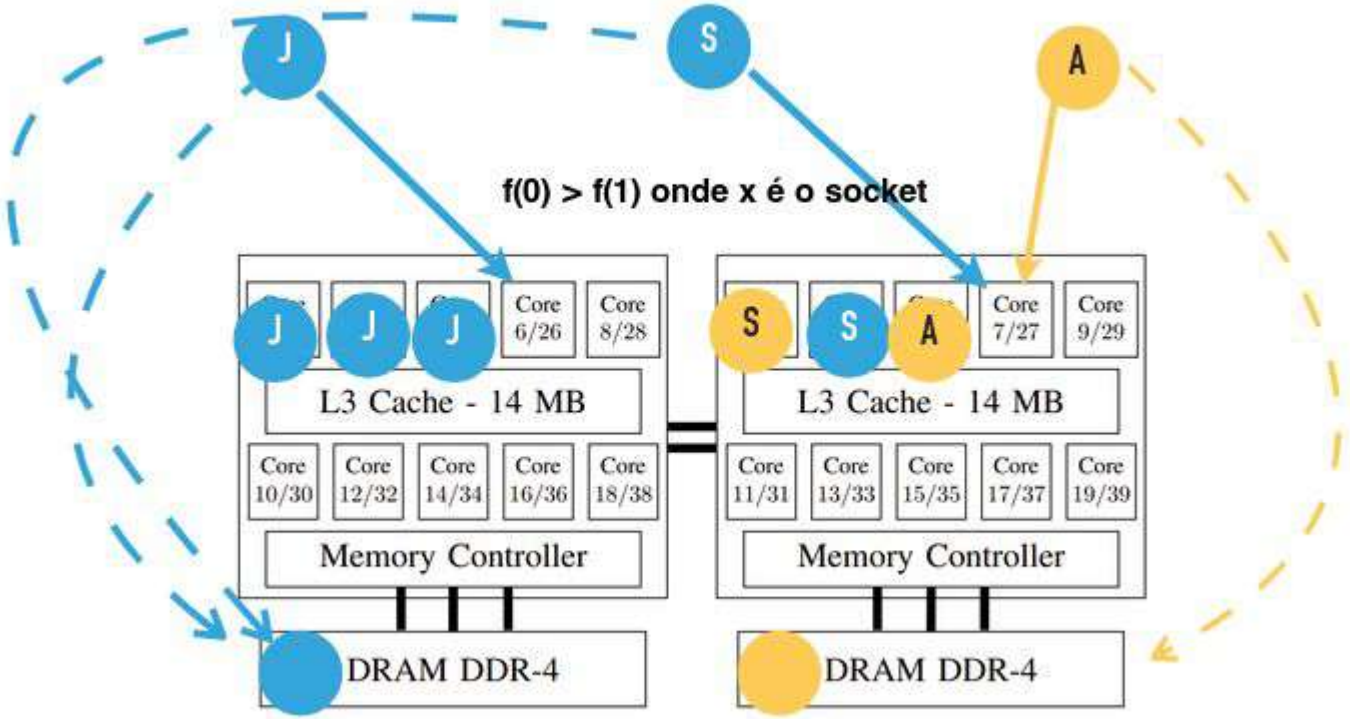
Threads como agente tomador de decisão



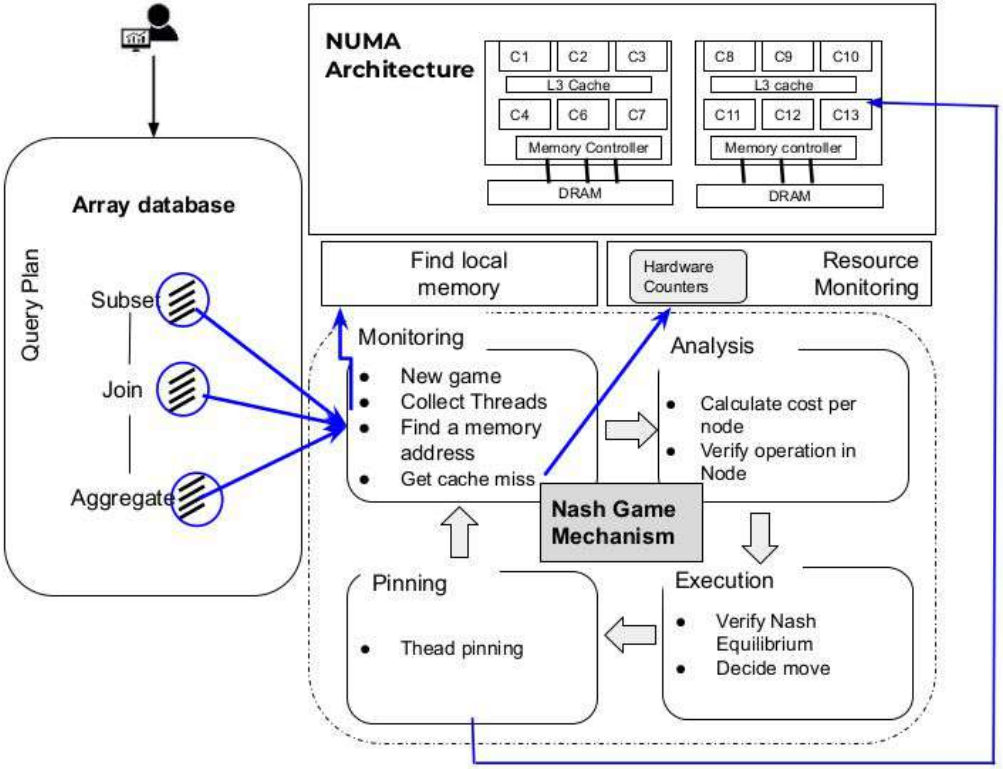
Threads como agente tomador de decisão



Threads como agente tomador de decisão



Threads como agente tomador de decisão – Mecanismo



Análise Experimental

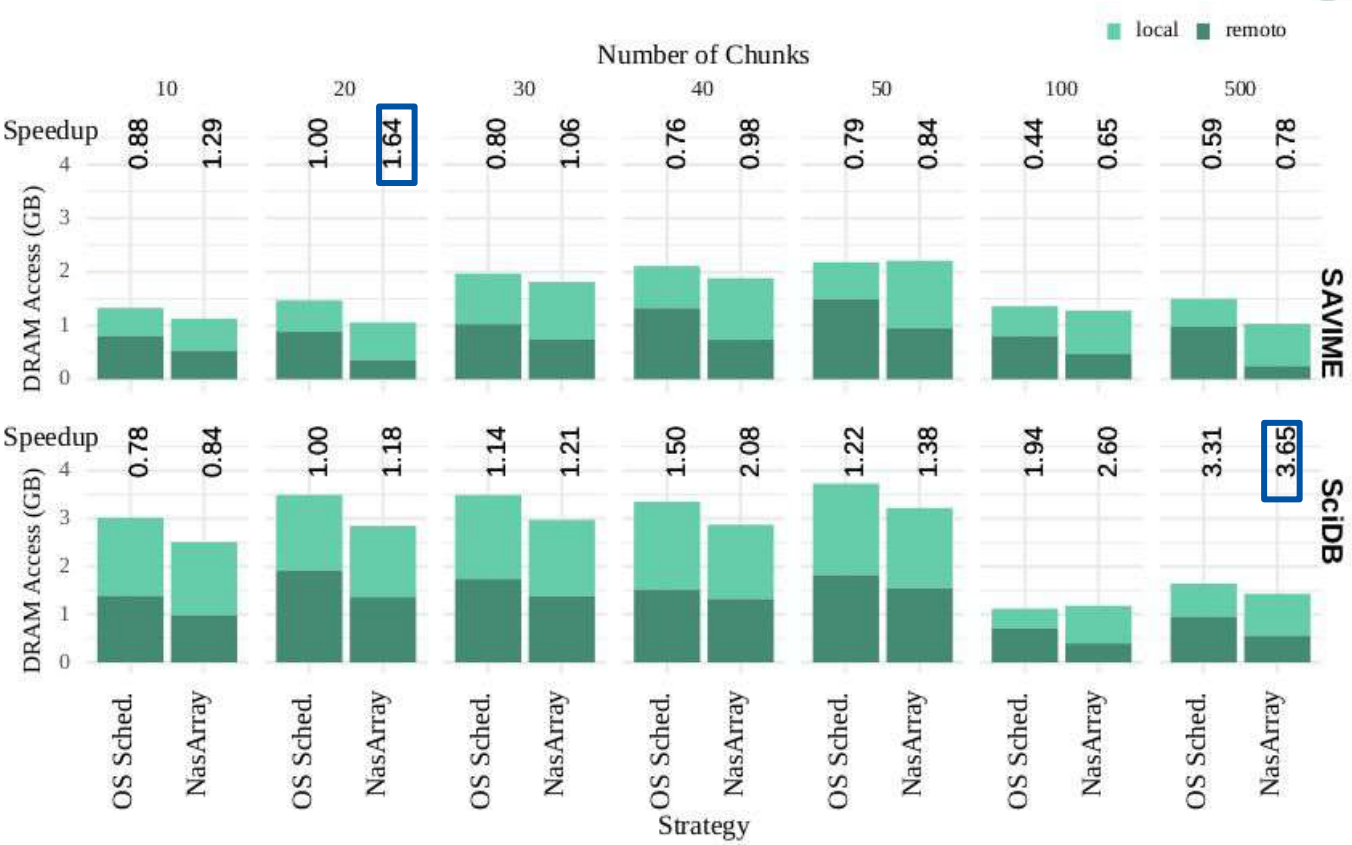
Ambiente de Testes

- Savime e SciDB
- 2-nós Intel Xeon Silver 4114 (20 núcleos), 128 GB DDR4 -- NUMA - Skylake
  - Ubuntu 18.04.01 LTS “Bionic Beaver” (Savime)
  - Ubuntu 14.04.6 LTS “Trusty Tahr”;
- 2-nós Intel Xeon E5 — 2630 (12 núcleos), 48 GB DDR3 -- NUMA - SandyBridge
  - CentOS 7.9.2009.

  1GB: quantidades diferentes de chunks
- 50 GB: Seletividade

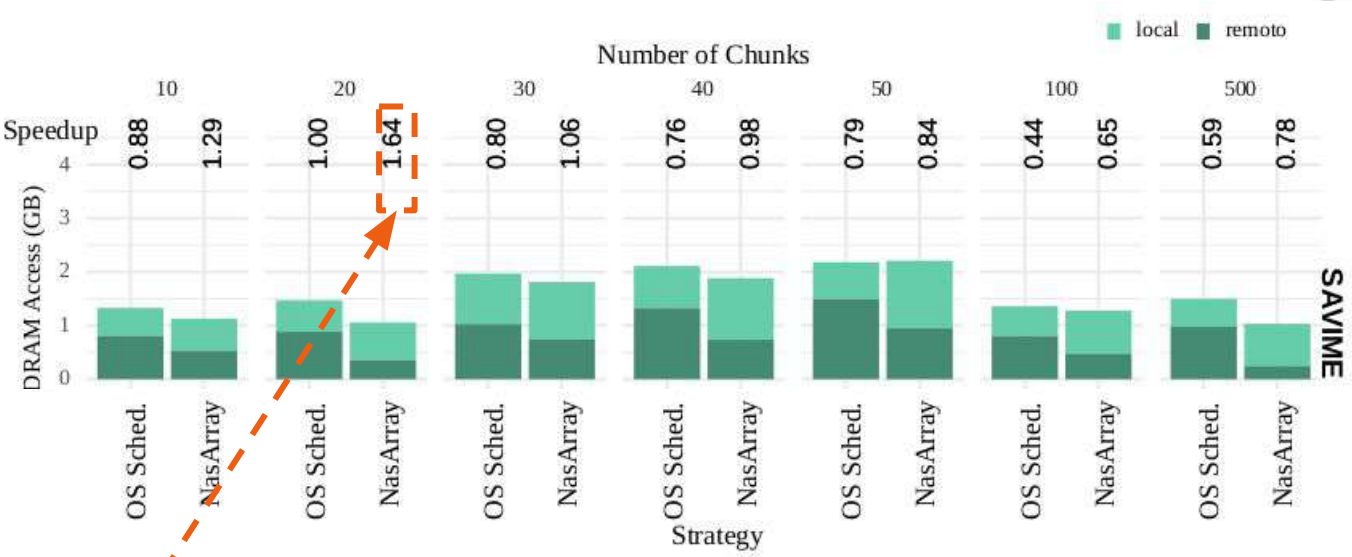
Impacto do acesso remoto a memória e Speedup — **subarray** — 1 GB

Melhor



Impacto do acesso remoto a memória e Speedup – **subarray** – 1 GB

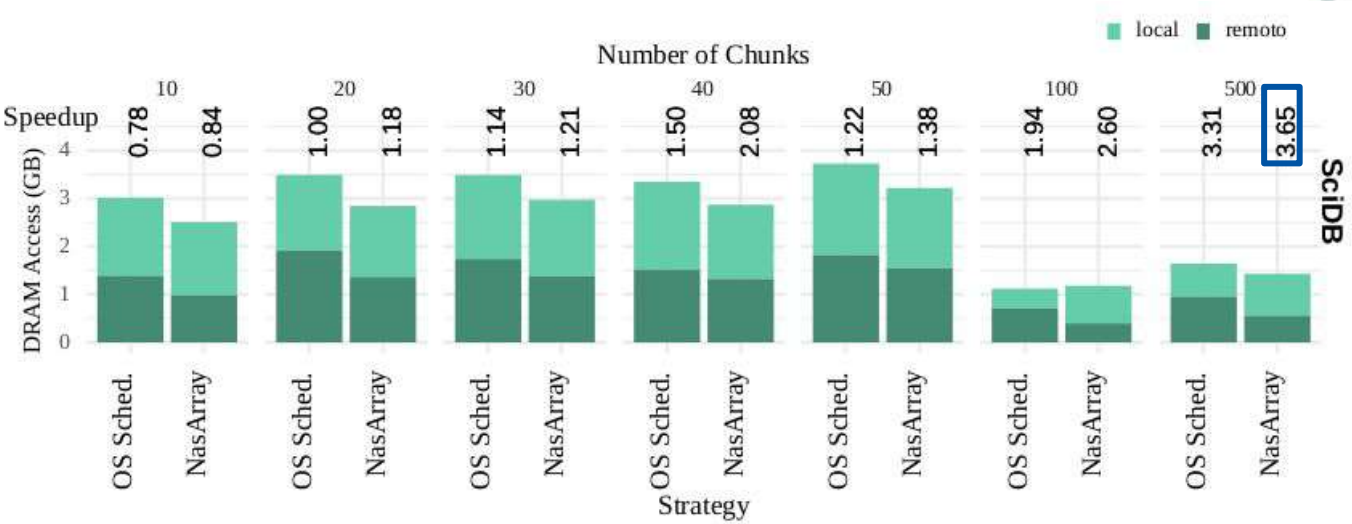
Melhor



- O número de chunks influencia o desempenho
- SAVIME: menor número de chunks maior desempenho
- **2,46x** menos acessos remotos com o NasArray

Impacto do acesso remoto a memória e Speedup — **subarray** — 1 GB

Melhor



- SciDB é melhor com um número maior de chunks
- NasArray reduziu **1,71x** acessos remotos

Sumarização dos resultados agregação – 1 GB

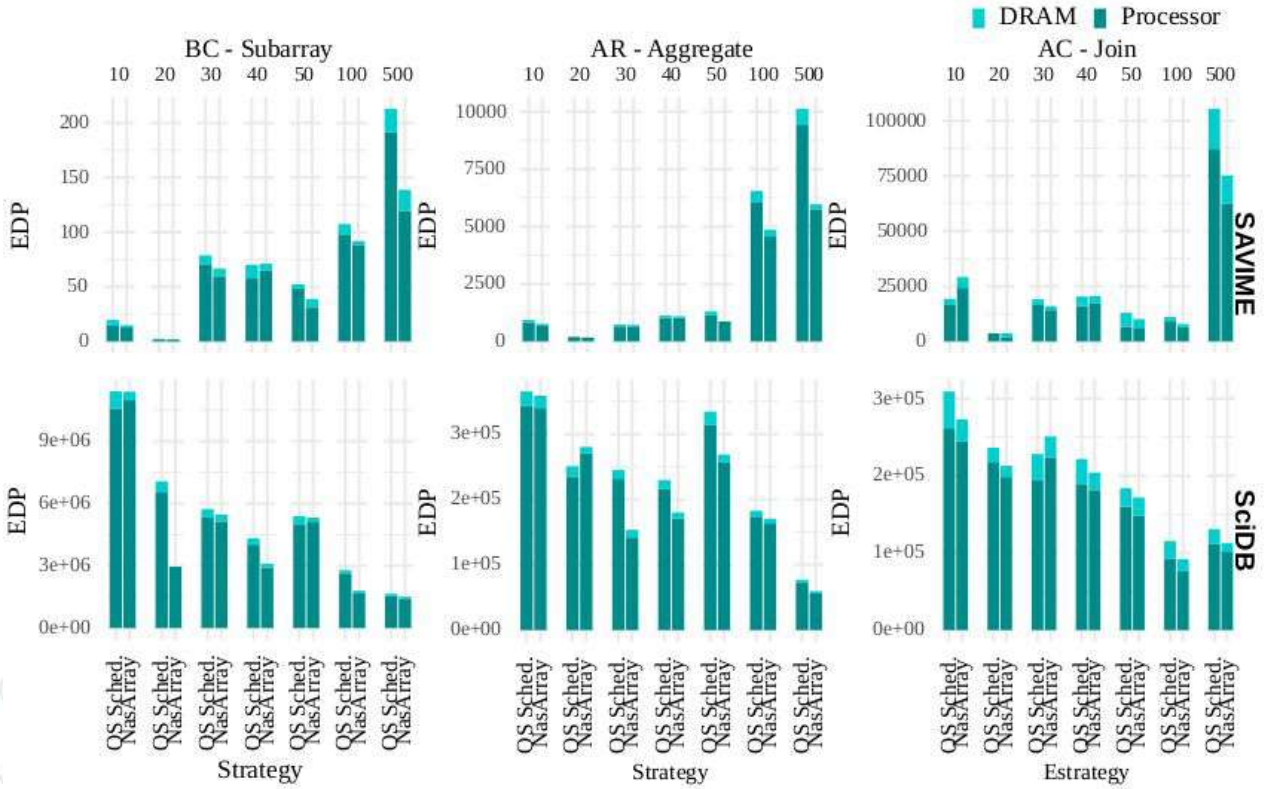
- SAVIME
  - Média de speedup: 1,64x
  - Média de redução de acessos remotos: 43%
- SCIDB
  - Média de speedup: 1,13x
  - Média de redução de acessos remotos: 22%

Sumarização dos resultados junção – 1 GB

- SAVIME
  - Média de speedup: 1,18x
  - Média de redução de acessos remotos: 31%
- SCIDB
  - Média de speedup: 1,08x
  - Média de redução de acessos remotos: 20%

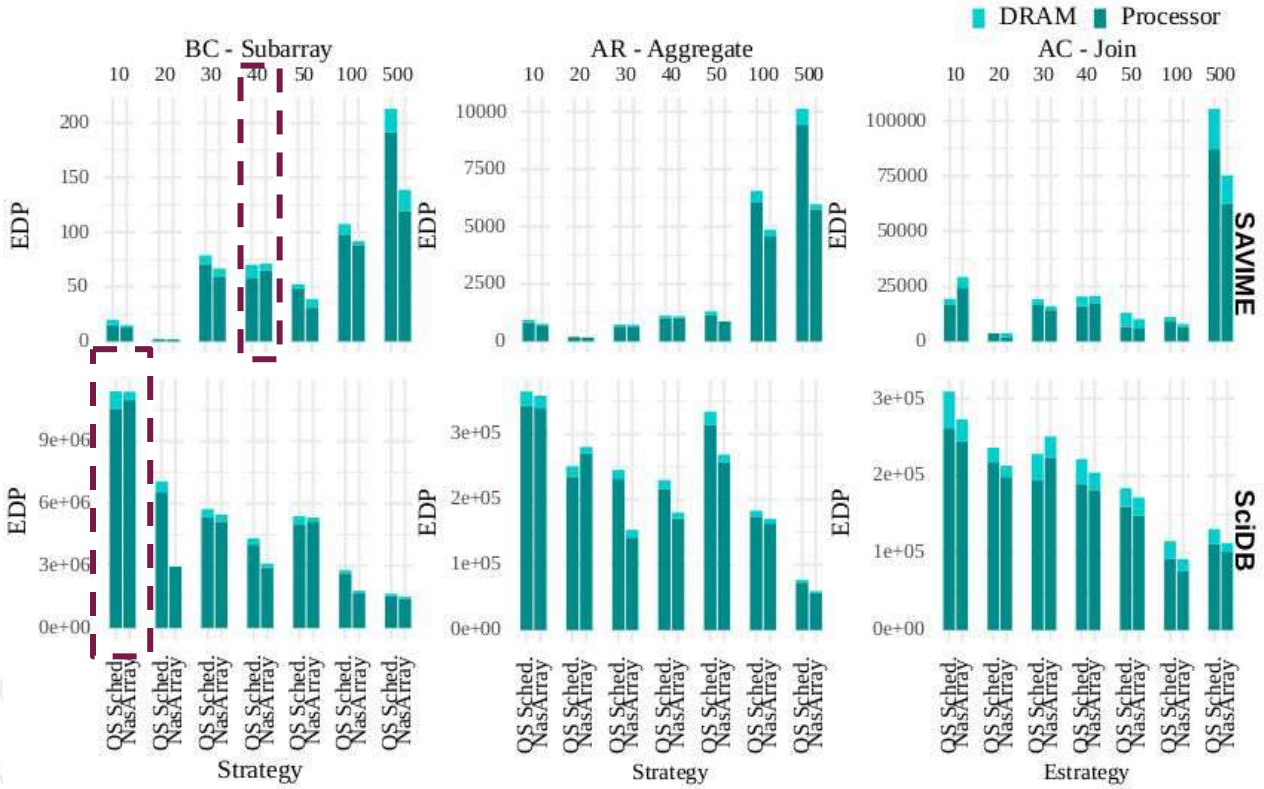
Eficiência energética – 1 GB EDP – *Energy-Delay Product*

Melhor



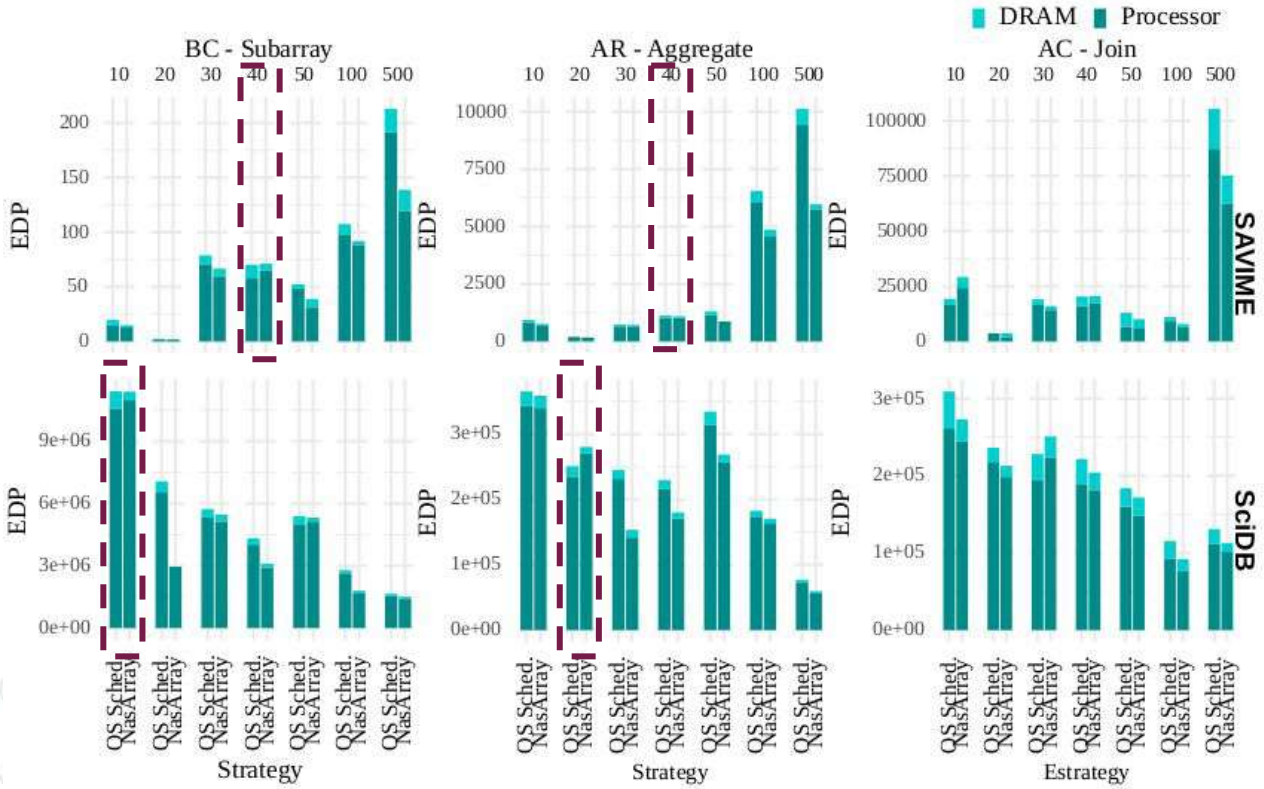
Eficiência energética – 1 GB EDP – *Energy-Delay Product*

Melhor



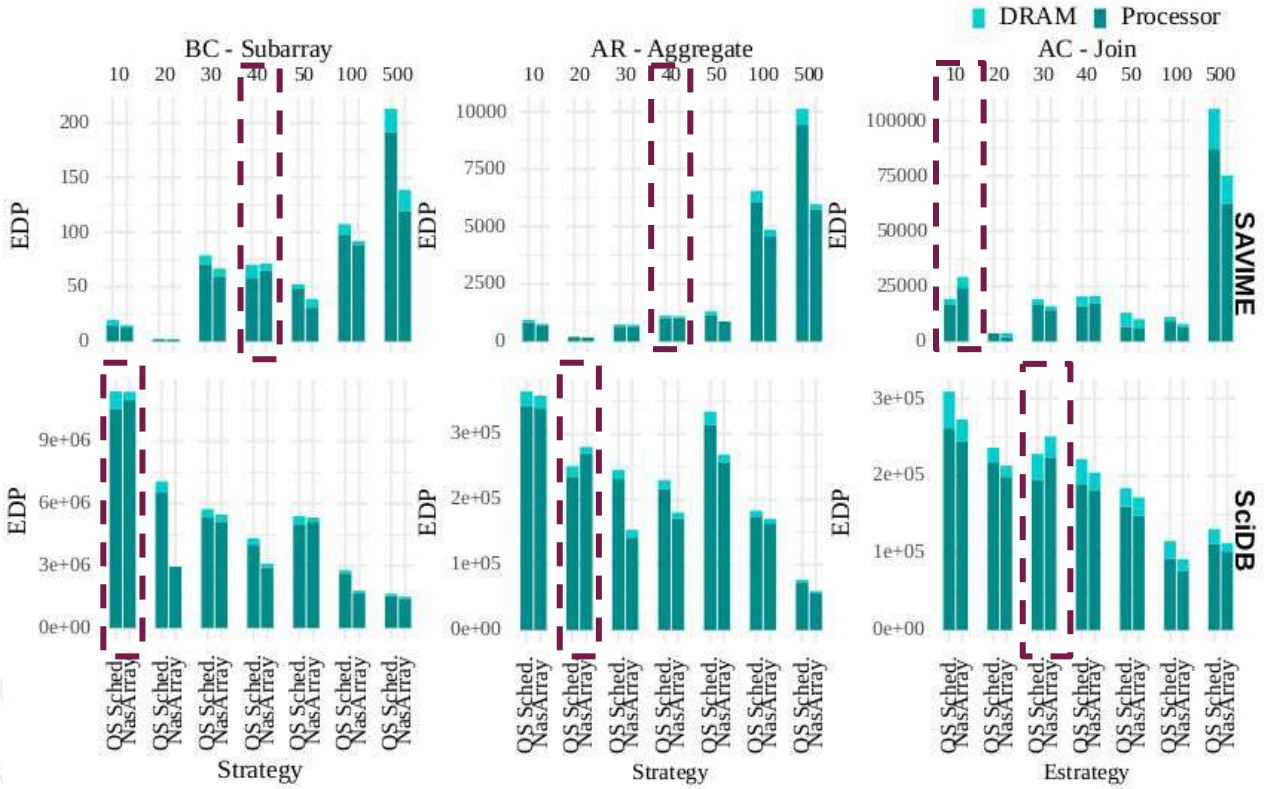
Eficiência energética – 1 GB EDP – *Energy-Delay Product*

Melhor



Eficiência energética – 1 GB EDP – *Energy-Delay Product*

Melhor



NasArray – dificuldade maior de encontrar o equilíbrio

Conclusão e trabalhos futuros

Conclusão – Relacional

- O agendamento de threads funciona melhor com menos núcleos
- (número ótimo local de núcleos) — Relacional
- Menos migrações de threads com menos movimentação de dados: até 3,87x de redução na taxa de tráfego
- Até 26% de economia de energia

Conclusões – matriz n-dimensional

- Nossos resultados confirmam que o NUMA afeta gravemente o desempenho da operação do subarray.
- O SO não reconhece o relacionamento entre os operadores de consulta no plano de execução da consulta
- Aceleração moderada com estratégias da literatura;
- As threads do SGBD multidimensional foram mapeadas no nó NUMA ideal para mitigar os acessos remotos a memória.

Trabalhos Futuros

- Cargas de trabalho mistas;
- Mecanismo de decisão entre fixação de threads e controle de recursos computacionais;
- Caracterização de outras operações do SGBD multidimensionais;
- Estudar o comportamento de SGBD multidimensionais em arquiteturas heterogêneas: CPU e GPU
- Arquiteturas Processamento em memória (PIM)
- Estudar melhorias necessárias em SGBD multidimensionais considerando a arquitetura dos clusters

Obrigado!

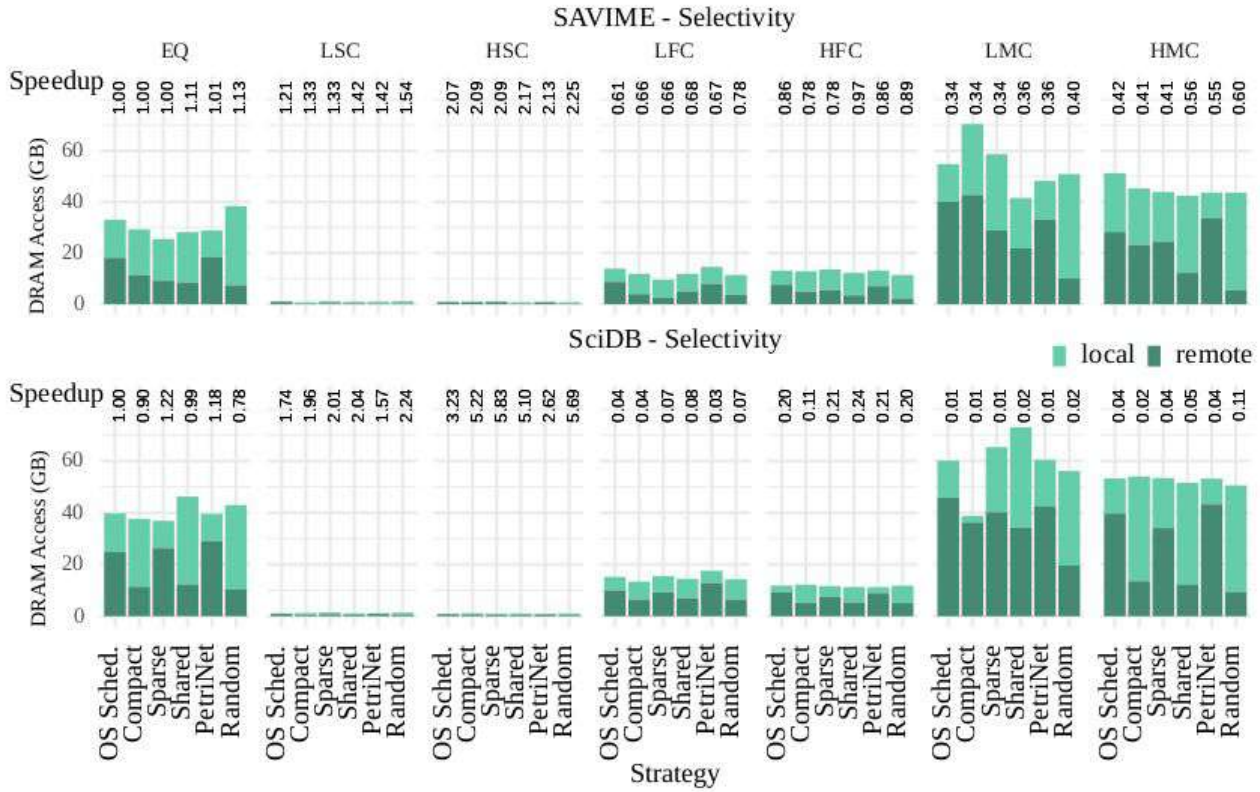
Perguntas?

sdominico@inf.ufpr.br

- On the performance limits of thread placement for Array Databases in Non-Uniform Memory Architectures. Simone Dominico, Marco Antonio Zanata Alves, Eduardo Cunha de Almeida; **Computing Journal, 2022.**
- Performance Analysis of Array Database Systems in Non-Uniform Memory Architecture. Simone Dominico, Marco Antonio Zanata Alves, Eduardo Cunha de Almeida, Jorge Augusto Meira; 29th **Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), 2021.**
- Multi-Core Allocation Model for Database Systems. Simone Dominico, Eduardo Cunha de Almeida; Proceedings of the **VLDB Ph.D. Workshop, 2018.**
- An Elastic Multi-Core Allocation Mechanism for Database Systems. Simone Dominico, Eduardo Cunha de Almeida, Jorge Augusto Meira, Marco Antonio Zanata Alves; **34th International Conference on Data Engineering (ICDE), 2018.**

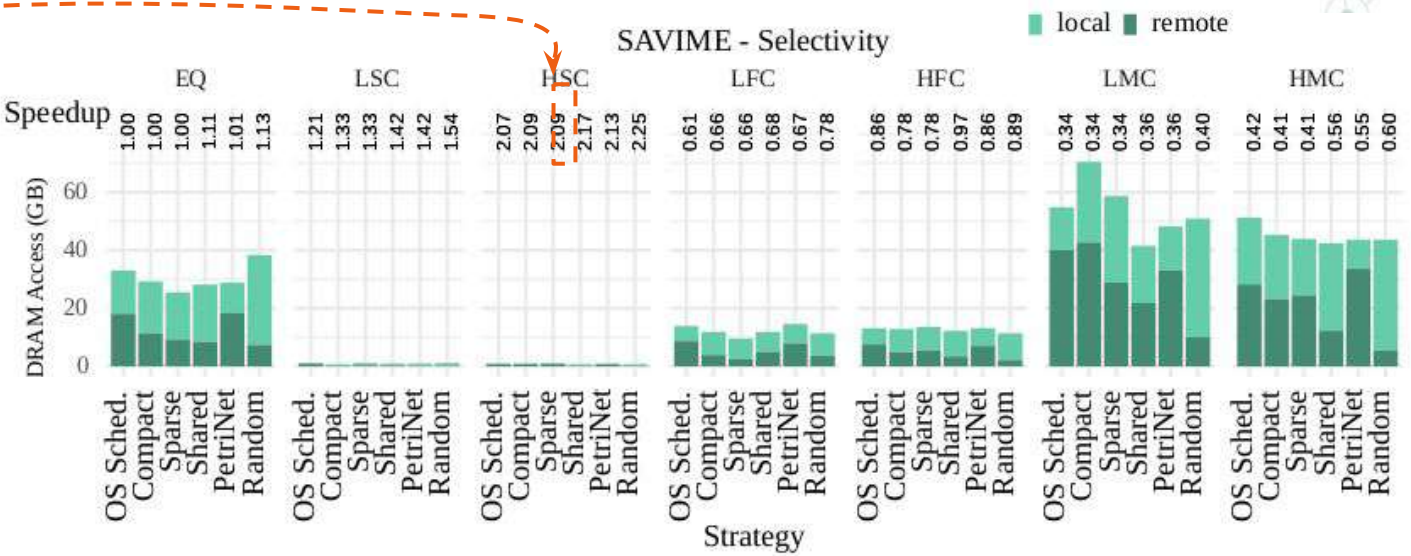
Comparação de performance operador subarray em uma base de dados de 50 GB

Melhor



Comparação de performance operador subarray em uma base de dados de 50 GB

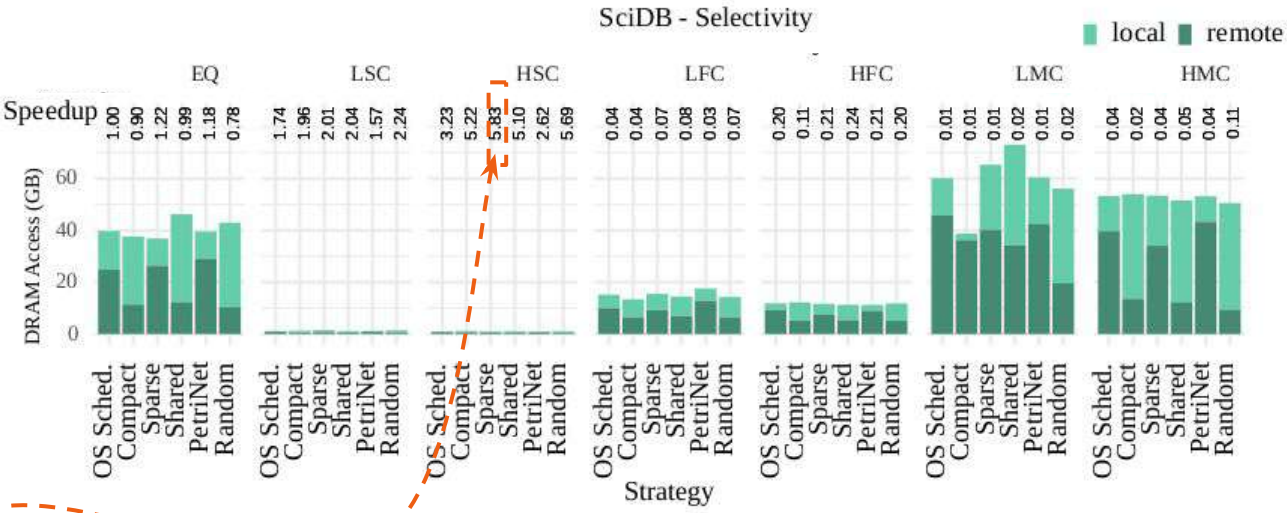
Melhor



- Random baixos acessos remotos menos **5x** acessos remotos
- 3,68% de pontencial de speedup
- Alta seletividade desempenho melhor

Comparação de performance operador subarray em uma base de dados de 50 GB

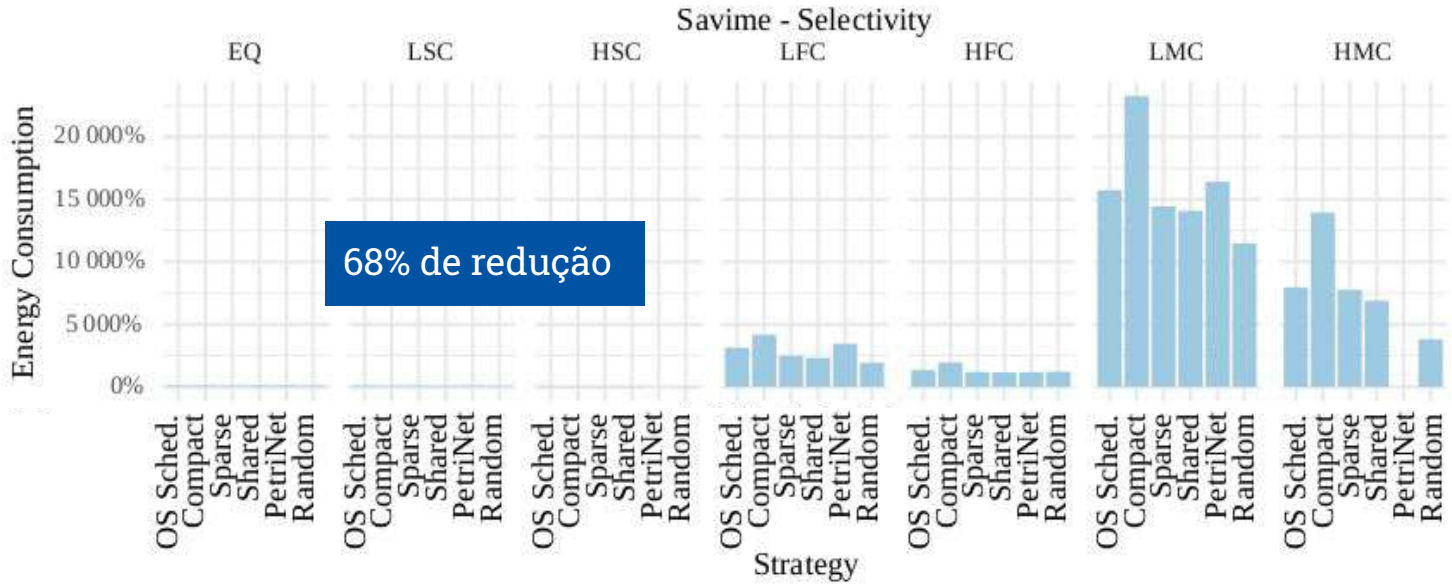
Melhor



- Random baixos acessos remotos menos **4,1x** acessos remotos
- Mapeamentos com resultados diferentes
- Sparse tem melhor desempenho

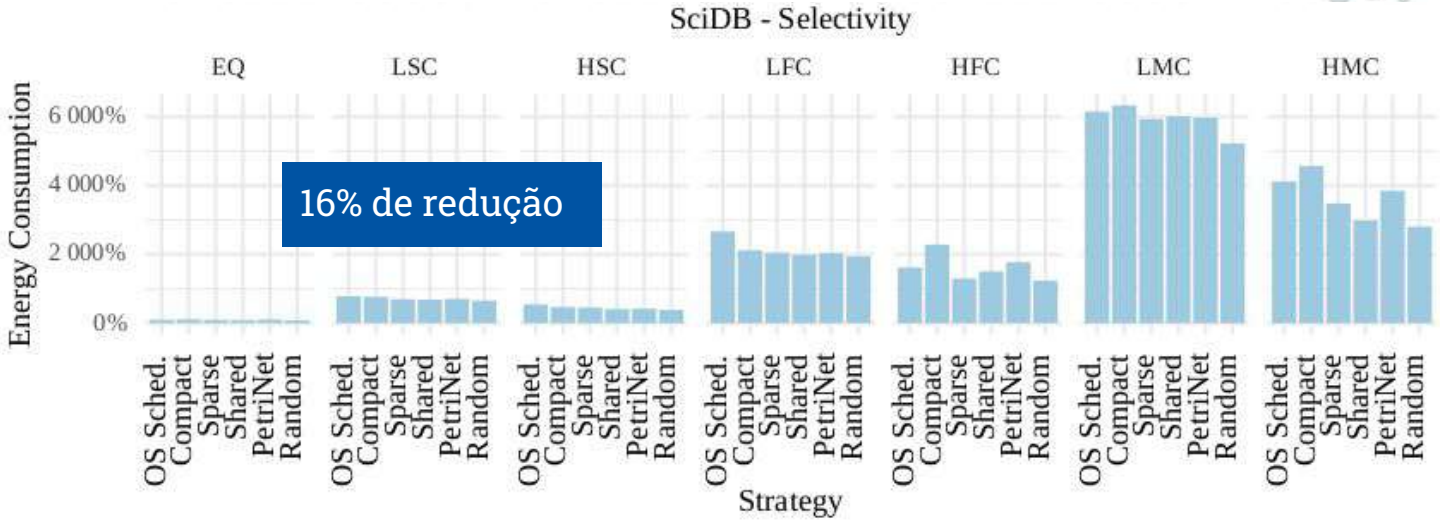
Consumo de energia operador subarray ema base de dados de 50 GB

Melhor

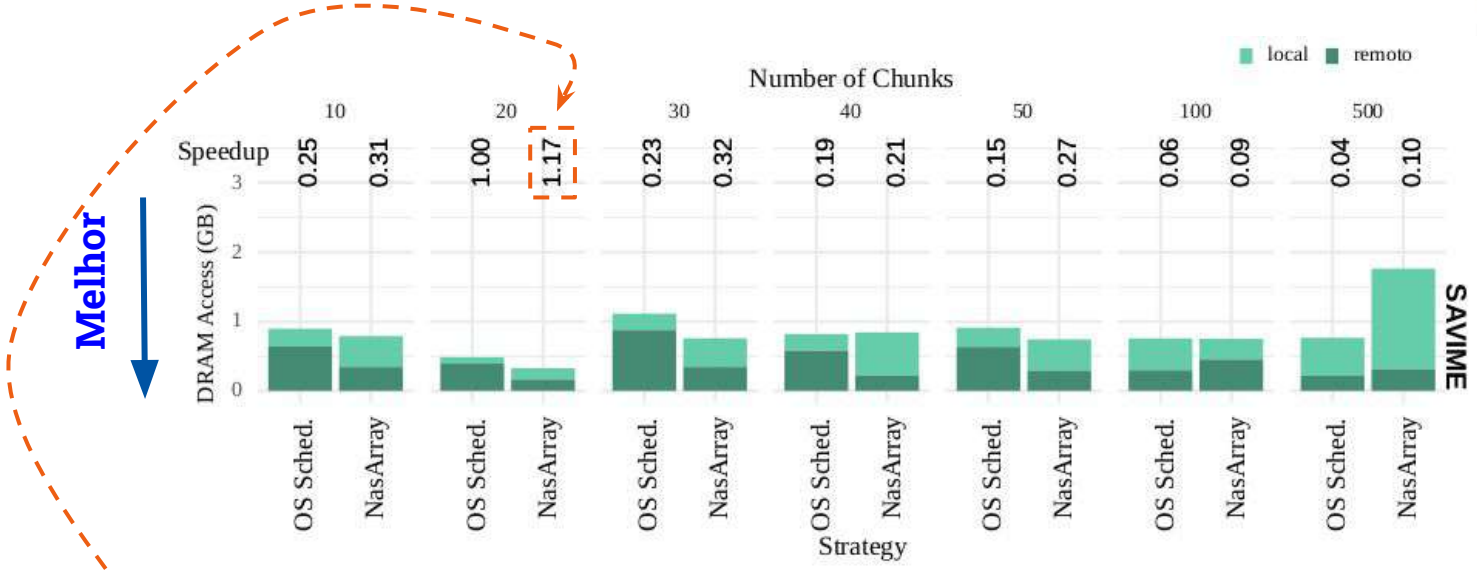


Consumo de energia operador subarray ema base de dados de 50 GB

Melhor

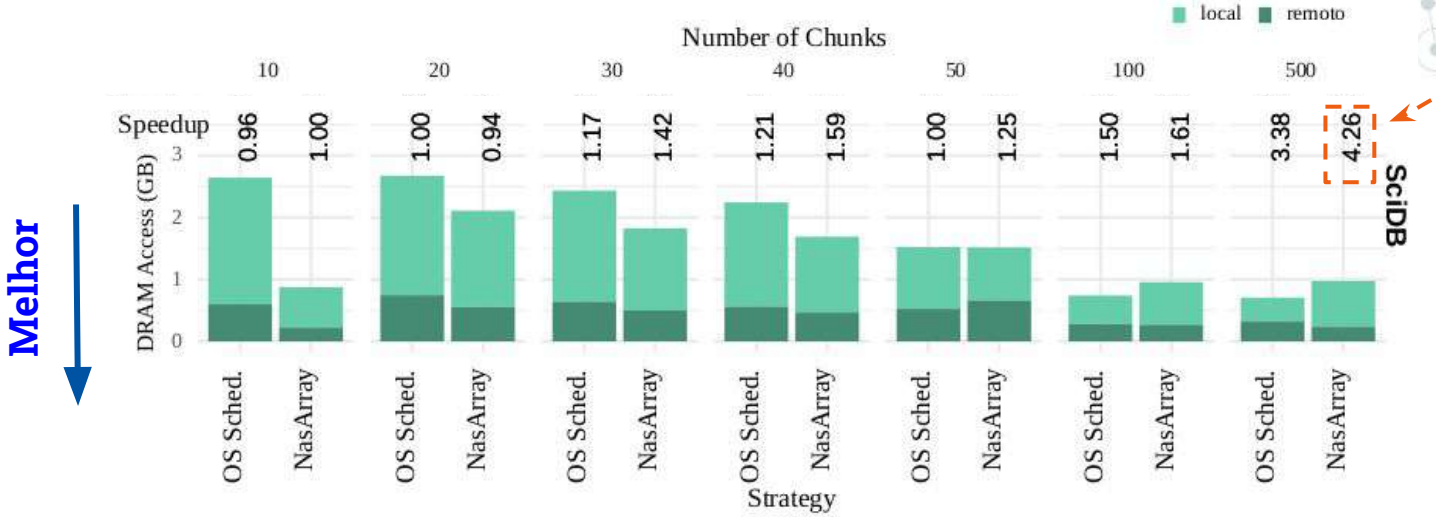


Impacto do acesso remoto a memória – **agregação** – 1 GB



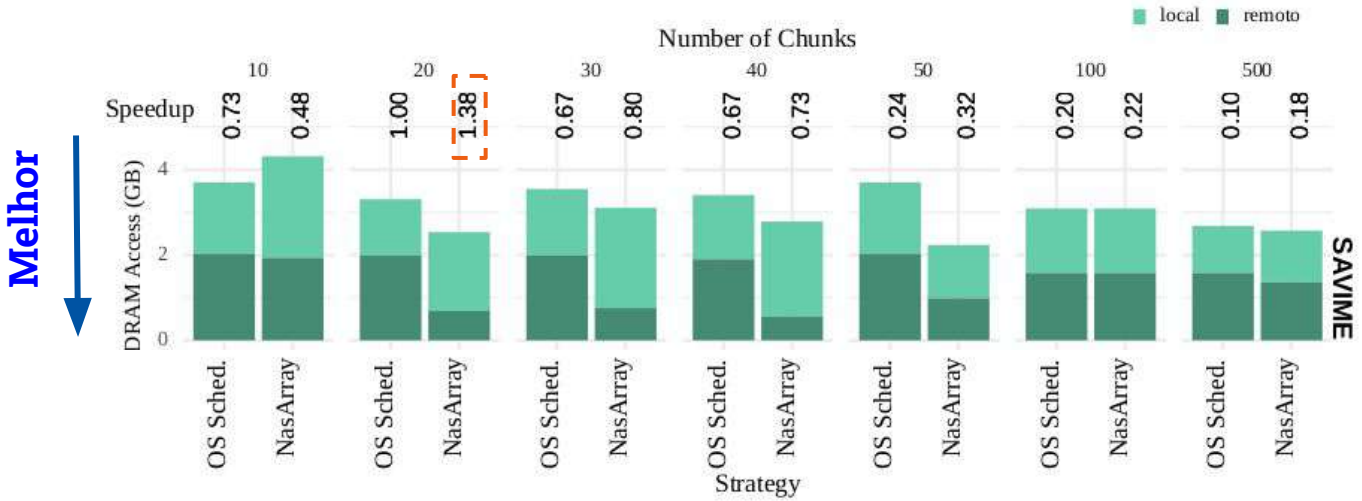
- SciDB é melhor com um número maior de chunks
- NasArray reduziu **1,71x** acessos remotos
- Agregação percorre todas as células analisadas

Impacto do acesso remoto a memória – **agregação** – 1 GB



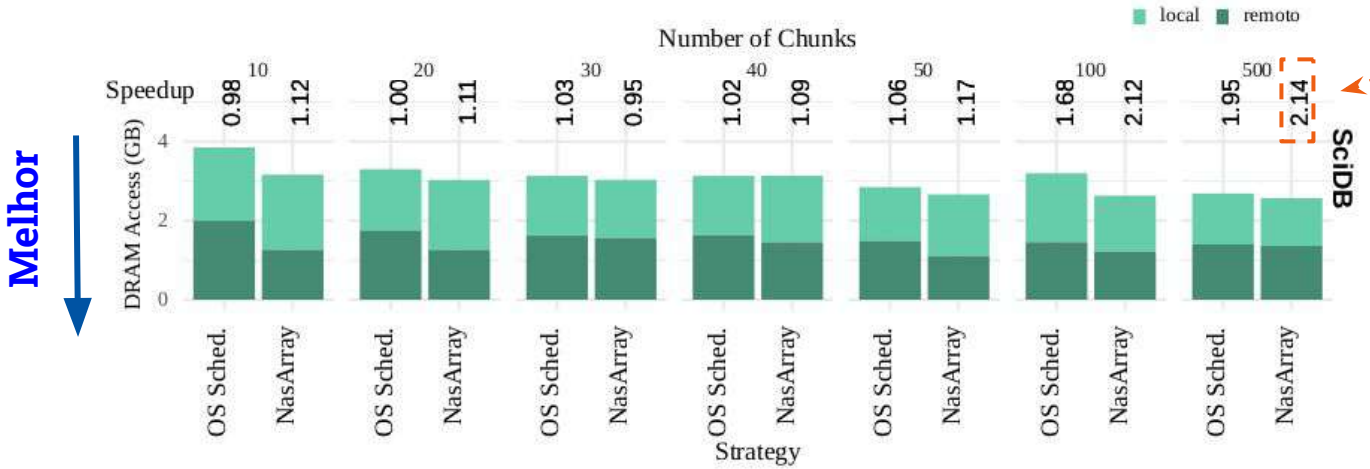
- SciDB é melhor com um número maior de chunks
- NasArray reduziu **1,2x** acessos remotos
- Acessa exatamente os valores utilizados

Impacto do acesso remoto a memória — **junção** — 1 GB



- NasArray reduziu **14% nos** acessos a memória
- 2.83x menos acesso remotos

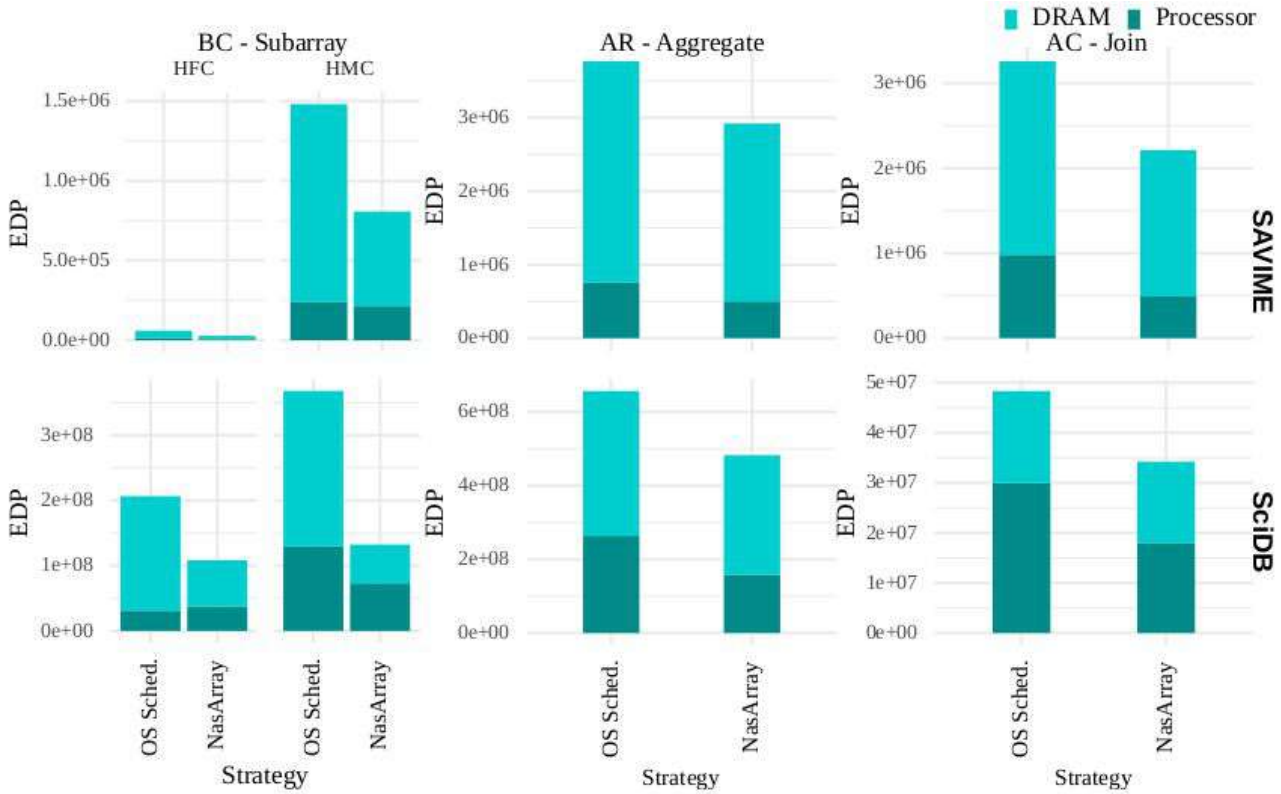
Impacto do acesso remoto a memória – **junção** – 1 GB



- NasArray reduziu **10%** nos acessos à memória
- Speedup **2,14**
- Percorre todas as dimensões

Eficiência energética — 50 GB

Melhor

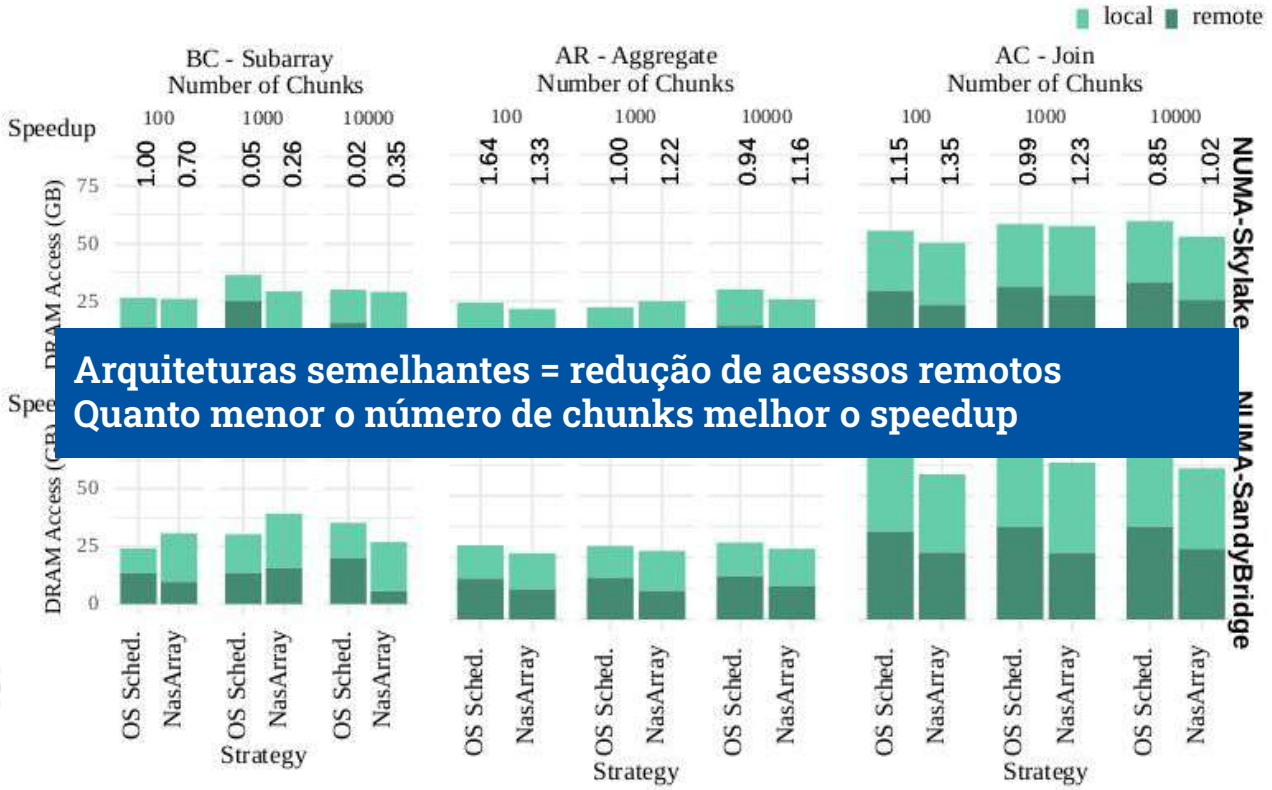


Comparação de duas arquiteturas -- SAVIME

Melhor

Comparação de duas arquiteturas -- SAVIME

Melhor



Comparação de duas arquiteturas -- SciDB

Melhor

Comparação de duas arquiteturas -- SciDB

Melhor

+acessos a memória