Multi-threaded Caching Problem

Wagner Frederic Denis Trystram Haifeng Xu

April 3, 2008

What to do on holiday?

Travel to some cities . . .

We have a database who can give you some advice.

▶ Chinese restaurant, sporting match	▶ Grenoble · · ·
▶ Italian food, sporting match · · ·	▶ Milan · · ·
▶ Chinese restaurant, a lot of people, shopping, culture, · · ·	▶ Paris · · ·

What to do on holiday?

Travel to some cities . . .

We have a database who can give you some advice.

- Chinese restaurant, sporting match
- Italian food, sporting match · · ·
- Chinese restaurant, a lot of people, shopping, culture, · · ·

- Grenoble · · ·
- Milan · · ·
- Paris · · ·

What to do on holiday?

Travel to some cities . . .

We have a database who can give you some advice.

- Chinese restaurant, sporting match
- Italian food, sporting match · · ·
- Chinese restaurant, a lot of people, shopping, culture, · · ·

- Grenoble · · ·
- Milan · · ·
- Paris · · ·

What to do on holiday?

Travel to some cities . . .

We have a database who can give you some advice.

- Chinese restaurant, sporting match
- Italian food, sporting match · · ·
- Chinese restaurant, a lot of people, shopping, culture, · · ·

- Grenoble · · ·
- Milan · · ·
- Paris · · ·

What to do on holiday?

Travel to some cities . . .

We have a database who can give you some advice.

- Chinese restaurant, sporting match
- Italian food, sporting match · · ·
- Chinese restaurant, a lot of people, shopping, culture, · · ·

- Grenoble · · ·
- Milan · · ·
- Paris · · ·

What to do on holiday?

Travel to some cities . . .

We have a database who can give you some advice.

▶ Chinese restaurant, sporting match	▶ Grenoble · · ·
▶ Italian food, sporting match · · ·	▶ Milan · · ·
▶ Chinese restaurant, a lot of people, shopping, culture, · · ·	▶ Paris · · ·

What to do on holiday?

Travel to some cities . . .

We have a database who can give you some advice.

- Chinese restaurant, sporting match
- Italian food, sporting match · · ·
- Chinese restaurant, a lot of people, shopping, culture, · · ·

- Grenoble · · ·
- Milan · · ·
- Paris · · ·

What to do on holiday?

Travel to some cities . . .

We have a database who can give you some advice.

▶ Chinese restaurant, sporting match	▶ Grenoble · · ·
▶ Italian food, sporting match · · ·	▶ Milan · · ·
▶ Chinese restaurant, a lot of people, shopping, culture, · · ·	▶ Paris · · ·

Multi-threaded Caching Problem

Moais

outline

Practical Problem: Hyper Project

To simplify . . .

Caching Problem

To Extend Caching Problem

Multi-threaded Caching Problem

Our Results for Special Case

To Be Continued . . .

Thanks

What to do on holiday?

Travel to some cities . . .

We have a database who can give you some advice.

- Chinese restaurant, sporting match
- Italian food, sporting match · · ·
- Chinese restaurant, a lot of people, shopping, culture, · · ·

- Grenoble · · ·
- Milan · · ·
- Paris · · ·

Summer 2008, welcome to Beijing!

What to do on holiday?

Travel to some cities . . .

We have a database who can give you some advice.

- Chinese restaurant, sporting match
- Italian food, sporting match ···
- Chinese restaurant, a lot of people, shopping, culture, ···

- Grenoble ···
- Milan ···
- Paris ···

In the view of database, how could I do better?

What to do on holiday?

Travel to some cities . . .

We have a database who can give you some advice.

- Chinese restaurant, sporting match
- Italian food, sporting match · · ·
- Chinese restaurant, a lot of people, shopping, culture, · · ·

- Grenoble · · ·
- Milan · · ·
- Paris · · ·

In the view of database, how could I do better?

What to do on holiday?

Travel to some cities . . .

We have a database who can give you some advice.

- Chinese restaurant, sporting match
- Italian food, sporting match · · ·
- Chinese restaurant, a lot of people, shopping, culture, · · ·

- Grenoble · · ·
- Milan · · ·
- Paris · · ·

In the view of database, how could I do better?

Outline

what to talk . . .

- A Model of Practical Problem: Hypercarte Project
- Caching Problem
- Multi-threaded Caching Problem
- To Be Continued

Outline

what to talk . . .

- A Model of Practical Problem: Hypercarte Project
- Caching Problem
- Multi-threaded Caching Problem
- To Be Continued

Outline

what to talk . . .

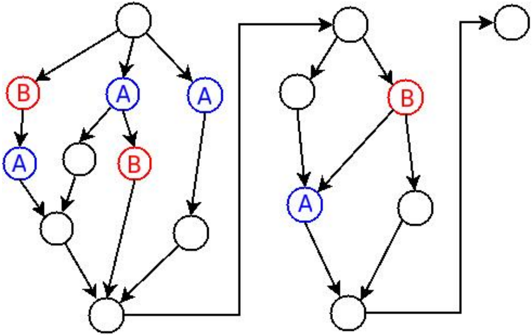
- A Model of Practical Problem: Hypercarte Project
- Caching Problem
- Multi-threaded Caching Problem
- To Be Continued

Outline

what to talk . . .

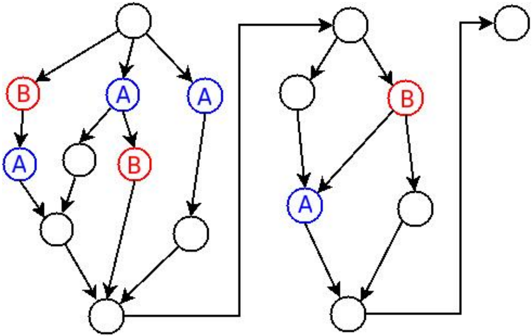
- A Model of Practical Problem: Hypercarte Project
- Caching Problem
- Multi-threaded Caching Problem
- To Be Continued

A model of Hyper Project



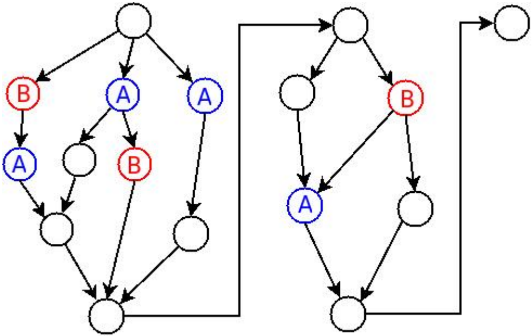
- Request is a DAG(Directed Acyclic Graph)
- m parallel machines
- Objective: C_{max} (Scheduling Problem)
- Some of them request the same task
- Store the results of some tasks?

A model of Hyper Project



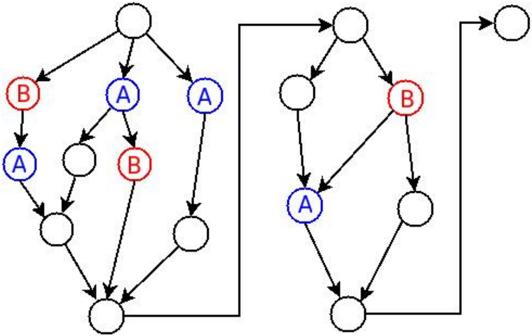
- Request is a DAG(Directed Acyclic Graph)
- m parallel machines
- Objective: C_{max} (Scheduling Problem)
- Some of them request the same task
- Store the results of some tasks?

A model of Hyper Project



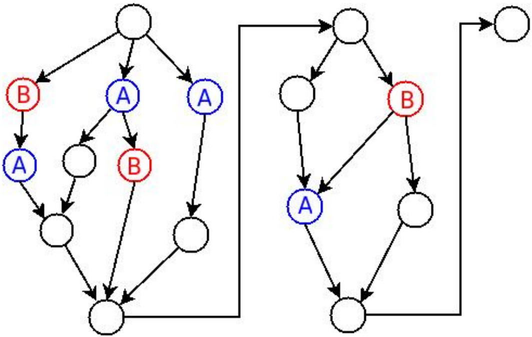
- Request is a DAG(Directed Acyclic Graph)
- m parallel machines
- Objective: C_{max} (Scheduling Problem)
- Some of them request the same task
- Store the results of some tasks?

A model of Hyper Project



- Request is a DAG(Directed Acyclic Graph)
- m parallel machines
- Objective: C_{max} (Scheduling Problem)
- Some of them request the same task
- Store the results of some tasks?

A model of Hyper Project



- Request is a DAG(Directed Acyclic Graph)
- m parallel machines
- Objective: C_{max} (Scheduling Problem)
- Some of them request the same task
- Store the results of some tasks?

Simplification of the Original Problem

Because original scheduling problem is hard, so we simplified it a bit . . .

▶ DAG	▶ one chain
▶ m machines	▶ one machine
▶ C_{max}	▶ C_{max}
▶ Cache	▶ Cache

Thus, we get caching problem.

Simplification of the Original Problem

Because original scheduling problem is hard, so we simplified it a bit . . .

- DAG
- m machines
- C_{max}
- Cache

- one chain
- one machine
- C_{max}
- Cache

Thus, we get caching problem.

Simplification of the Original Problem

Because original scheduling problem is hard, so we simplified it a bit . . .

- DAG
- m machines
- C_{max}
- Cache

- one chain
- one machine
- C_{max}
- Cache

Thus, we get caching problem.

Simplification of the Original Problem

Because original scheduling problem is hard, so we simplified it a bit . . .

- DAG
- m machines
- C_{max}
- Cache

- one chain
- one machine
- C_{max}
- Cache

Thus, we get caching problem.

Simplification of the Original Problem

Because original scheduling problem is hard, so we simplified it a bit . . .

- DAG
- m machines
- C_{max}
- Cache

- one chain
- one machine
- C_{max}
- Cache

Thus, we get caching problem.

Simplification of the Original Problem

Because original scheduling problem is hard, so we simplified it a bit . . .

▶ DAG	▶ one chain
▶ m machines	▶ one machine
▶ C_{max}	▶ C_{max}
▶ Cache	▶ Cache

Thus, we get caching problem.

Simplification of the Original Problem

Because original scheduling problem is hard, so we simplified it a bit . . .

- DAG
- m machines
- C_{max}
- Cache

- one chain
- one machine
- C_{max}
- Cache

Thus, we get caching problem.

Simplification of the Original Problem

Because original scheduling problem is hard, so we simplified it a bit . . .

- DAG
- m machines
- C_{max}
- Cache

- one chain
- one machine
- C_{max}
- Cache

Thus, we get caching problem.

Description of Caching Problem

$\{T_1, T_2, \ldots, T_L\}$

Cache: K

$g : N \rightarrow \{1, \ldots, L\}$

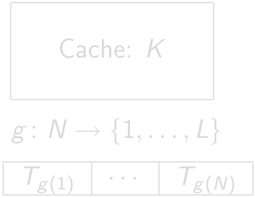
$T_{g(1)}$	$\cdots$	$T_{g(N)}$

Input

- A set of tasks
  - processing time: P_i
  - size of result: S_i
- One processor
- A cache of capacity K
  - $\sum_{T_i \in Cache} S_i \leq K$
- Request chain: g

Description of Caching Problem

$\{T_1, T_2, \ldots, T_L\}$



Cache: K

$g : N \rightarrow \{1, \ldots, L\}$

$T_{g(1)}$ | $\cdots$ | $T_{g(N)}$

Input

- A set of tasks
  - processing time: P_i
  - size of result: S_i
- One processor
- A cache of capacity K
  - $\sum_{T_i \in Cache} S_i \leq K$
- Request chain: g

Description of Caching Problem

$\{T_1, T_2, \ldots, T_L\}$

Cache: K

$g : N \rightarrow \{1, \ldots, L\}$

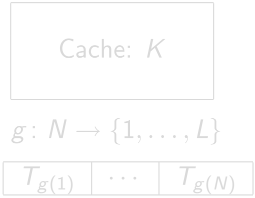
$T_{g(1)}$	$\cdots$	$T_{g(N)}$

Input

- A set of tasks
  - processing time: P_i
  - size of result: S_i
- One processor
- A cache of capacity K
  - $\sum_{T_i \in Cache} S_i \leq K$
- Request chain: g

Description of Caching Problem

$\{T_1, T_2, \ldots, T_L\}$



Cache: K

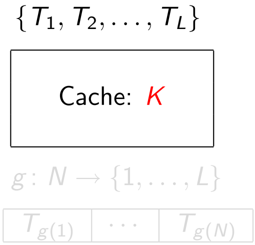
$g : N \rightarrow \{1, \ldots, L\}$

$T_{g(1)}$	$\cdots$	$T_{g(N)}$

Input

- A set of tasks
  - processing time: P_i
  - size of result: S_i
- One processor
- A cache of capacity K
  - $\sum_{T_i \in Cache} S_i \leq K$
- Request chain: g

Description of Caching Problem



$\{T_1, T_2, \dots, T_L\}$

Cache: K

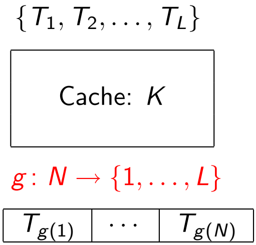
$g: N \to \{1, \dots, L\}$

$T_{g(1)}$ | $\cdots$ | $T_{g(N)}$

Input

- A set of tasks
  - processing time: P_i
  - size of result: S_i
- One processor
- A cache of capacity K
  - $\sum_{T_i \in Cache} S_i \leq K$
- Request chain: g

Description of Caching Problem



$\{T_1, T_2, \dots, T_L\}$

Cache: K

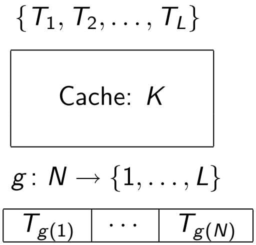
$g : N \rightarrow \{1, \dots, L\}$

$T_{g(1)}$ | $\cdots$ | $T_{g(N)}$

Input

- A set of tasks
  - processing time: P_i
  - size of result: S_i
- One processor
- A cache of capacity K
  - $\sum_{T_i \in Cache} S_i \leq K$
- Request chain: g

Description of Caching Problem



$\{T_1, T_2, \ldots, T_L\}$

Cache: K

$g : N \rightarrow \{1, \ldots, L\}$

$T_{g(1)}$	$\cdots$	$T_{g(N)}$

Input

- A set of tasks
  - processing time: P_i
  - size of result: S_i
- One processor
- A cache of capacity K
  - $\sum_{T_i \in Cache} S_i \leq K$
- Request chain: g

$$\min : C_{max} = \sum_{i=1}^{N} P_{g(i)} \times X_{g(i)}$$

Description of Caching Problem

$\{T_1, T_2, \ldots, T_L\}$

Cache: K

$g : N \rightarrow \{1, \ldots, L\}$

$T_{g(1)}$	$\cdots$	$T_{g(N)}$

Input

- A set of tasks
  - processing time: P_i
  - size of result: S_i
- One processor
- A cache of capacity K
  - $\sum_{T_i \in Cache} S_i \leq K$
- Request chain: g

$$\min : C_{max} = \sum_{i=1}^{N} P_{g(i)} \times X_{g(i)}$$

$$X_{g(i)} = \begin{cases} 0 & \text{if task } T_{g(i)} \text{ is in the cache in the } i_{th} \text{ iteration} \\ 1 & \text{otherwise} \end{cases}$$

An Example of Caching Problem

TASK	SIZE	TIME
A	1	2
B	1	1
C	2	1

We have a cache of capacity 2

A	B	C	A	B	C	C	C	C	B

processing time:

cache:

An Example of Caching Problem

TASK	SIZE	TIME
A	1	2
B	1	1
C	2	1

We have a cache of capacity 2

A	B	C	A	B	C	C	C	C	B

processing time:

cache:

An Example of Caching Problem

TASK	SIZE	TIME
A	1	2
B	1	1
C	2	1

We have a cache of capacity 2

A	B	C	A	B	C	C	C	C	B

processing time:

cache:

An Example of Caching Problem

TASK	SIZE	TIME
A	1	2
B	1	1
C	2	1

We have a cache of capacity 2

A	B	C	A	B	C	C	C	C	B

processing time:

cache:

A

P_A

An Example of Caching Problem

TASK	SIZE	TIME
A	1	2
B	1	1
C	2	1

We have a cache of capacity 2

A	B	C	A	B	C	C	C	C	B

processing time:

cache:

A B

$P_A + P_B$

An Example of Caching Problem

TASK	SIZE	TIME
A	1	2
B	1	1
C	2	1

We have a cache of capacity 2

A	B	C	A	B	C	C	C	C	B

processing time:

cache:

A B

$$P_A + P_B + P_C$$

An Example of Caching Problem

TASK	SIZE	TIME
A	1	2
B	1	1
C	2	1

We have a cache of capacity 2

A	B	C	A	B	C	C	C	C	B

processing time:

$$P_A + P_B + P_C$$

cache:

A B

An Example of Caching Problem

TASK	SIZE	TIME
A	1	2
B	1	1
C	2	1

We have a cache of capacity 2

A	B	C	A	B	C	C	C	C	B

processing time:

cache:

A B

$$P_A + P_B + P_C$$

An Example of Caching Problem

TASK	SIZE	TIME
A	1	2
B	1	1
C	2	1

We have a cache of capacity 2

A	B	C	A	B	C	C	C	C	B

processing time:

$$P_A + P_B + P_C + P_C$$

cache:

C

An Example of Caching Problem

TASK	SIZE	TIME
A	1	2
B	1	1
C	2	1

We have a cache of capacity 2

A	B	C	A	B	C	C	C	C	B

processing time:

$$P_A + P_B + P_C + P_C$$

cache:

C

An Example of Caching Problem

TASK	SIZE	TIME
A	1	2
B	1	1
C	2	1

We have a cache of capacity 2

A	B	C	A	B	C	C	C	C	B

processing time:

cache:

C

$$P_A + P_B + P_C + P_C$$

An Example of Caching Problem

TASK	SIZE	TIME
A	1	2
B	1	1
C	2	1

We have a cache of capacity 2

A	B	C	A	B	C	C	C	C	B

processing time:

cache:

C

$$P_A + P_B + P_C + P_C$$

An Example of Caching Problem

TASK	SIZE	TIME
A	1	2
B	1	1
C	2	1

We have a cache of capacity 2

A	B	C	A	B	C	C	C	C	B

processing time:

$$P_A + P_B + P_C + P_C + P_B$$

cache:

C

An Example of Caching Problem

TASK	SIZE	TIME
A	1	2
B	1	1
C	2	1

We have a cache of capacity 2

A	B	C	A	B	C	C	C	C	B

cache:

C

processing time:

$$P_A + P_B + P_C + P_C + P_B$$

we save:

$P_A + P_B + 3P_C$

Complexity of Caching Problem

Previous Results

The complexity depends on *the processing time* and *the size of results*.

	SIZE	TIME	Complexity
Cost Model	1	$\mathbb{Z}^+$	P
Fault Model	$\mathbb{Z}^+$	1	?
General Model	$\mathbb{Z}^+$	$\mathbb{Z}^+$	NP-hard

Complexity of Caching Problem

Previous Results

The complexity depends on *the processing time* and *the size of results*.

	SIZE	TIME	Complexity
Cost Model	1	$\mathbb{Z}^+$	P
Fault Model	$\mathbb{Z}^+$	1	?
General Model	$\mathbb{Z}^+$	$\mathbb{Z}^+$	NP-hard

Complexity of Caching Problem

Previous Results

The complexity depends on *the processing time* and *the size of results*.

	SIZE	TIME	Complexity
Cost Model	1	$\mathbb{Z}^+$	P
Fault Model	$\mathbb{Z}^+$	1	?
General Model	$\mathbb{Z}^+$	$\mathbb{Z}^+$	NP-hard

Complexity of Caching Problem

Previous Results

The complexity depends on *the processing time* and *the size of results*.

	SIZE	TIME	Complexity
Cost Model	1	$\mathbb{Z}^+$	P
Fault Model	$\mathbb{Z}^+$	1	?
General Model	$\mathbb{Z}^+$	$\mathbb{Z}^+$	NP-hard

Extend Caching Problem

Caching problem is a little far away from our original model, so we extend caching problem a bit.

We extend the number of request chain.

- DAG
- m machines
- C_{max}
- Cache

- ONE chain
- one machines
- C_{max}
- Cache

- Several Chains
- one machines
- C_{max}
- Cache

Extend Caching Problem

Caching problem is a little far away from our original model, so we extend caching problem a bit.

We extend the number of request chain.

- DAG
- m machines
- C_{max}
- Cache

- ONE chain
- one machines
- C_{max}
- Cache

- Several Chains
- one machines
- C_{max}
- Cache

Extend Caching Problem

Caching problem is a little far away from our original model, so we extend caching problem a bit.

We extend the number of request chain.

- DAG
- m machines
- C_{max}
- Cache

- ONE chain
- one machines
- C_{max}
- Cache

- Several Chains
- one machines
- C_{max}
- Cache

Extend Caching Problem

Caching problem is a little far away from our original model, so we extend caching problem a bit.

We extend the number of request chain.

- DAG
- m machines
- C_{max}
- Cache

- ONE chain
- one machines
- C_{max}
- Cache

- Several Chains
- one machines
- C_{max}
- Cache

Extend Caching Problem

Caching problem is a little far away from our original model, so we extend caching problem a bit.

We extend the number of request chain.

- DAG
- m machines
- C_{max}
- Cache

- ONE chain
- one machines
- C_{max}
- Cache

- Several Chains
- one machines
- C_{max}
- Cache

Description of Multi-threaded Caching Problem

Cache of capacity K

$$S_{task} = \{T_1, \ldots, T_L\}$$

$$g : Q \times N \rightarrow \{1, \ldots, L\}$$

$T_{g(1,1)}$ $T_{g(1,2)}$ $T_{g(1,3)}$ $T_{g(1,N_1)}$

$T_{g(2,1)}$ $T_{g(2,2)}$ $T_{g(2,N_2)}$

$\vdots$

$T_{g(Q,1)}$ $T_{g(Q,2)}$ $T_{g(Q,3)}$ $T_{g(Q,N_Q)}$

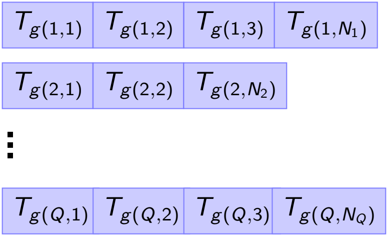
- Which chain should be served each iteration?
- Whether to store the result after serving it?

Description of Multi-threaded Caching Problem

Cache of capacity K

$$S_{task} = \{T_1, \ldots, T_L\}$$

$$g: Q \times N \to \{1, \ldots, L\}$$



$T_{g(1,1)}$ $T_{g(1,2)}$ $T_{g(1,3)}$ $T_{g(1,N_1)}$

$T_{g(2,1)}$ $T_{g(2,2)}$ $T_{g(2,N_2)}$

$\vdots$

$T_{g(Q,1)}$ $T_{g(Q,2)}$ $T_{g(Q,3)}$ $T_{g(Q,N_Q)}$

- Which chain should be served each iteration?
- Whether to store the result after serving it?

Description of Multi-threaded Caching Problem

$$\boxed{\text{Cache of capacity } K} \qquad S_{task} = \{T_1, \ldots, T_L\}$$

$$g : Q \times N \rightarrow \{1, \ldots, L\}$$

$$\boxed{T_{g(1,1)}}\boxed{T_{g(1,2)}}\boxed{T_{g(1,3)}}\boxed{T_{g(1,N_1)}}$$

$$\boxed{T_{g(2,1)}}\boxed{T_{g(2,2)}}\boxed{T_{g(2,N_2)}}$$

$$\vdots$$

$$\boxed{T_{g(Q,1)}}\boxed{T_{g(Q,2)}}\boxed{T_{g(Q,3)}}\boxed{T_{g(Q,N_Q)}}$$

- Which chain should be served each iteration?
- Whether to store the result after serving it?

Description of Multi-threaded Caching Problem

Cache of capacity K

$$S_{task} = \{T_1, \ldots, T_L\}$$

$$g : Q \times N \rightarrow \{1, \ldots, L\}$$

$T_{g(1,1)}$ $T_{g(1,2)}$ $T_{g(1,3)}$ $T_{g(1,N_1)}$

$T_{g(2,1)}$ $T_{g(2,2)}$ $T_{g(2,N_2)}$

$\vdots$

$T_{g(Q,1)}$ $T_{g(Q,2)}$ $T_{g(Q,3)}$ $T_{g(Q,N_Q)}$

- Which chain should be served each iteration?
- Whether to store the result after serving it?

Multi-threaded Caching Problem

Moais

outline

Practical Problem: Hyper Project

To simplify . . .

Caching Problem

To Extend Caching Problem

Multi-threaded Caching Problem

Our Results for Special Case

To Be Continued . . .

Thanks

Algorithm for Multi-threaded Caching Problem

Idea

$T_{g(1,1)}$ $T_{g(1,2)}$ $T_{g(1,3)}$ $T_{g(1,Y_1)}$ $T_{g(1,N_1)}$

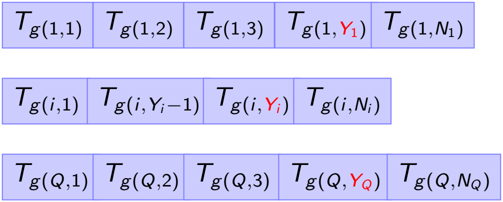
$T_{g(i,1)}$ $T_{g(i,Y_i-1)}$ $T_{g(i,Y_i)}$ $T_{g(i,N_i)}$

$T_{g(Q,1)}$ $T_{g(Q,2)}$ $T_{g(Q,3)}$ $T_{g(Q,Y_Q)}$ $T_{g(Q,N_Q)}$

- At position $\overrightarrow{Y} = [Y_1, Y_2, \cdots, Y_Q] \in \prod_{i=1}^{Q} N_i$
- S_Y is the set of tasks appearing before $\overrightarrow{Y}$
- Dynamic programming:
  For all $\overrightarrow{Y}$ and $F \subseteq S_Y$ with $|F| \leq K$, denote by $OPT(\overrightarrow{Y} \| F)$ the minimum processing time at position $\overrightarrow{Y}$ with F in the cache.

Algorithm for Multi-threaded Caching Problem

Idea



$T_{g(1,1)}$ $T_{g(1,2)}$ $T_{g(1,3)}$ $T_{g(1,Y_1)}$ $T_{g(1,N_1)}$

$T_{g(i,1)}$ $T_{g(i,Y_i-1)}$ $T_{g(i,Y_i)}$ $T_{g(i,N_i)}$

$T_{g(Q,1)}$ $T_{g(Q,2)}$ $T_{g(Q,3)}$ $T_{g(Q,Y_Q)}$ $T_{g(Q,N_Q)}$

- At position $\overrightarrow{Y} = [Y_1, Y_2, \cdots, Y_Q] \in \prod_{i=1}^{Q} N_i$
- S_Y is the set of tasks appearing before $\overrightarrow{Y}$
- Dynamic programming:
  For all $\overrightarrow{Y}$ and $F \subseteq S_Y$ with $|F| \leq K$, denote by $OPT(\overrightarrow{Y} \| F)$ the minimum processing time at position $\overrightarrow{Y}$ with F in the cache.

Algorithm for Multi-threaded Caching Problem

Idea

$T_{g(1,1)}$ | $T_{g(1,2)}$ | $T_{g(1,3)}$ | $T_{g(1,Y_1)}$ | $T_{g(1,N_1)}$

$T_{g(i,1)}$ | $T_{g(i,Y_i-1)}$ | $T_{g(i,Y_i)}$ | $T_{g(i,N_i)}$

$T_{g(Q,1)}$ | $T_{g(Q,2)}$ | $T_{g(Q,3)}$ | $T_{g(Q,Y_Q)}$ | $T_{g(Q,N_Q)}$

- At position $\overrightarrow{Y} = [Y_1, Y_2, \cdots, Y_Q] \in \prod_{i=1}^{Q} N_i$
- S_Y is the set of tasks appearing before $\overrightarrow{Y}$
- Dynamic programming:
  For all $\overrightarrow{Y}$ and $F \subseteq S_Y$ with $|F| \leq K$, denote by $OPT(\overrightarrow{Y} \| F)$ the minimum processing time at position $\overrightarrow{Y}$ with F in the cache.

Algorithm for Multi-threaded Caching Problem

Idea

$T_{g(1,1)}$	$T_{g(1,2)}$	$T_{g(1,3)}$	$T_{g(1,Y_1)}$	$T_{g(1,N_1)}$
$T_{g(i,1)}$	$T_{g(i,Y_i-1)}$	$T_{g(i,Y_i)}$	$T_{g(i,N_i)}$	
$T_{g(Q,1)}$	$T_{g(Q,2)}$	$T_{g(Q,3)}$	$T_{g(Q,Y_Q)}$	$T_{g(Q,N_Q)}$

- At position $\overrightarrow{Y} = [Y_1, Y_2, \cdots, Y_Q] \in \prod_{i=1}^{Q} N_i$
- S_Y is the set of tasks appearing before $\overrightarrow{Y}$
- Dynamic programming:
  For all $\overrightarrow{Y}$ and $F \subseteq S_Y$ with $|F| \leq K$, denote by $OPT(\overrightarrow{Y} \| F)$ the minimum processing time at position $\overrightarrow{Y}$ with F in the cache.

Algorithm for Multi-threaded Caching Problem

$T_{g(1,1)}$ | $T_{g(1,2)}$ | $T_{g(1,3)}$ | $T_{g(1,Y_1)}$ | $T_{g(1,N_1)}$

$T_{g(i,1)}$ | $T_{g(i,Y_i-1)}$ | $T_{g(i,Y_i)}$ | $T_{g(i,N_i)}$

$T_{g(Q,1)}$ | $T_{g(Q,2)}$ | $T_{g(Q,3)}$ | $T_{g(Q,Y_Q)}$ | $T_{g(Q,N_Q)}$

How to calculate $OPT(\overrightarrow{Y}\|F)$?

Suppose we have $OPT(\overrightarrow{Z}\|F')$ for all $\overrightarrow{Z}$ before $\overrightarrow{Y}$ and $F' \subseteq S_Z$ with $|F'| \leq K$.

If we go backwards one step from position $\overrightarrow{Y}$, we have at most Q possibilities, say $[Y_1, \cdots, Y_Q] - [0, \cdot\cdot, 1, \cdot\cdot, 0]^i$.

Assuming we go backwards one step at the i_{th} chain, arriving at position $\overrightarrow{Z}$, for all F' such that $|F' \setminus F| \leq 1$, we go forwards to calculate $\min\{OPT(\overrightarrow{Y}\|F')\}$.

Algorithm for Multi-threaded Caching Problem

$T_{g(1,1)}$	$T_{g(1,2)}$	$T_{g(1,3)}$	$T_{g(1,Y_1)}$	$T_{g(1,N_1)}$
$T_{g(i,1)}$	$T_{g(i,Y_i-1)}$	$T_{g(i,Y_i)}$	$T_{g(i,N_i)}$	
$T_{g(Q,1)}$	$T_{g(Q,2)}$	$T_{g(Q,3)}$	$T_{g(Q,Y_Q)}$	$T_{g(Q,N_Q)}$

How to calculate $OPT(\overrightarrow{Y}\|F)$?
Suppose we have $OPT(\overrightarrow{Z}\|F')$ for all $\overrightarrow{Z}$ before $\overrightarrow{Y}$ and $F' \subseteq S_Z$ with $|F'| \leq K$.

If we go backwards one step from position $\overrightarrow{Y}$, we have at most Q possibilities, say $[Y_1, \cdots, Y_Q] - [0, \cdot\cdot, 1, \cdot\cdot, 0]^i$.
Assuming we go backwards one step at the i_{th} chain, arriving at position $\overrightarrow{Z}$, for all F' such that $|F' \backslash F| \leq 1$, we go forwards to calculate $\min\{OPT(\overrightarrow{Y}\|F')\}$.

Algorithm for Multi-threaded Caching Problem

$T_{g(1,1)}$	$T_{g(1,2)}$	$T_{g(1,3)}$	$T_{g(1,Y_1)}$	$T_{g(1,N_1)}$

$T_{g(i,1)}$	$T_{g(i,Y_i-1)}$	$T_{g(i,Y_i)}$	$T_{g(i,N_i)}$

$T_{g(Q,1)}$	$T_{g(Q,2)}$	$T_{g(Q,3)}$	$T_{g(Q,Y_Q)}$	$T_{g(Q,N_Q)}$

How to calculate $OPT(\overrightarrow{Y} \| F)$?

Suppose we have $OPT(\overrightarrow{Z} \| F')$ for all $\overrightarrow{Z}$ before $\overrightarrow{Y}$ and $F' \subseteq S_Z$ with $|F'| \leq K$.

If we go backwards one step from position $\overrightarrow{Y}$, we have at most Q possibilities, say $[Y_1, \cdots, Y_Q] - [0, \cdot\cdot, 1, \cdot\cdot, 0]^i$.

Assuming we go backwards one step at the i_{th} chain, arriving at position $\overrightarrow{Z}$, for all F' such that $|F' \backslash F| \leq 1$, we go forwards to calculate $\min\{OPT(\overrightarrow{Y} \| F')\}$.

Algorithm for Multi-threaded Caching Problem

$T_{g(1,1)}$	$T_{g(1,2)}$	$T_{g(1,3)}$	$T_{g(1,Y_1)}$	$T_{g(1,N_1)}$
$T_{g(i,1)}$	$T_{g(i,Y_i-1)}$	$T_{g(i,Y_i)}$	$T_{g(i,N_i)}$	
$T_{g(Q,1)}$	$T_{g(Q,2)}$	$T_{g(Q,3)}$	$T_{g(Q,Y_Q)}$	$T_{g(Q,N_Q)}$

How to calculate $OPT(\overrightarrow{Y}\|F)$?

Suppose we have $OPT(\overrightarrow{Z}\|F')$ for all $\overrightarrow{Z}$ before $\overrightarrow{Y}$ and $F' \subseteq S_Z$ with $|F'| \leq K$.

If we go backwards one step from position $\overrightarrow{Y}$, we have at most Q possibilities, say $[Y_1, \cdots, Y_Q] - [0, \cdot\cdot, 1, \cdot\cdot, 0]^i$.

Assuming we go backwards one step at the i_{th} chain, arriving at position $\overrightarrow{Z}$, for all F' such that $|F' \backslash F| \leq 1$, we go forwards to calculate $\min\{OPT(\overrightarrow{Y}\|F')\}$.

Analysis of the Algorithm

$$T_{g(1,1)} \quad T_{g(1,2)} \quad T_{g(1,3)} \quad T_{g(1,Y_1)} \quad T_{g(1,N_1)}$$

$$T_{g(i,1)} \quad T_{g(i,Y_i-1)} \quad T_{g(i,Y_i)} \quad T_{g(i,N_i)}$$

$$T_{g(Q,1)} \quad T_{g(Q,2)} \quad T_{g(Q,3)} \quad T_{g(Q,Y_Q)} \quad T_{g(Q,N_Q)}$$

In fact, we consider all the possibilities of the optimal solution $OPT(\overrightarrow{Y}\|F)$.

The total number of combinatorics is $\prod_{j=1}^{Q} N_j \times \binom{L}{K}$.

To calculate ever one of the function, we have Q choices in each iteration. In each iteration we consider all the set F' which is different from F at most one task, so we have at most L comparison

Exponential algorithm:

$$O(\ Q \times L \times \prod_{j=1}^{Q} N_j \times \binom{L}{K}\ )$$

Analysis of the Algorithm

$T_{g(1,1)}$	$T_{g(1,2)}$	$T_{g(1,3)}$	$T_{g(1,Y_1)}$	$T_{g(1,N_1)}$

$T_{g(i,1)}$	$T_{g(i,Y_i-1)}$	$T_{g(i,Y_i)}$	$T_{g(i,N_i)}$

$T_{g(Q,1)}$	$T_{g(Q,2)}$	$T_{g(Q,3)}$	$T_{g(Q,Y_Q)}$	$T_{g(Q,N_Q)}$

In fact, we consider all the possibilities of the optimal solution $OPT(\overrightarrow{Y} \| F)$.
The total number of combinatorics is $\prod_{j=1}^{Q} N_j \times \binom{L}{K}$.

To calculate ever one of the function, we have Q choices in each iteration. In each iteration we consider all the set F' which is different from F at most one task, so we have at most L comparison

Exponential algorithm:

$$O(\ Q \times L \times \prod_{j=1}^{Q} N_j \times \binom{L}{K}\ )$$

Analysis of the Algorithm

$$\boxed{T_{g(1,1)}}\boxed{T_{g(1,2)}}\boxed{T_{g(1,3)}}\boxed{T_{g(1,Y_1)}}\boxed{T_{g(1,N_1)}}$$

$$\boxed{T_{g(i,1)}}\boxed{T_{g(i,Y_i-1)}}\boxed{T_{g(i,Y_i)}}\boxed{T_{g(i,N_i)}}$$

$$\boxed{T_{g(Q,1)}}\boxed{T_{g(Q,2)}}\boxed{T_{g(Q,3)}}\boxed{T_{g(Q,Y_Q)}}\boxed{T_{g(Q,N_Q)}}$$

In fact, we consider all the possibilities of the optimal solution $OPT(\overrightarrow{Y}\|F)$.
The total number of combinatorics is $\prod_{j=1}^{Q} N_j \times \binom{L}{K}$.

To calculate ever one of the function, we have Q choices in each iteration. In each iteration we consider all the set F' which is different from F at most one task, so we have at most L comparison

Exponential algorithm:

$$O(\ Q \times L \times \prod_{j=1}^{Q} N_j \times \binom{L}{K}\ )$$

Analysis of the Algorithm

$T_{g(1,1)}$	$T_{g(1,2)}$	$T_{g(1,3)}$	$T_{g(1,Y_1)}$	$T_{g(1,N_1)}$

$T_{g(i,1)}$	$T_{g(i,Y_i-1)}$	$T_{g(i,Y_i)}$	$T_{g(i,N_i)}$

$T_{g(Q,1)}$	$T_{g(Q,2)}$	$T_{g(Q,3)}$	$T_{g(Q,Y_Q)}$	$T_{g(Q,N_Q)}$

In fact, we consider all the possibilities of the optimal solution $OPT(\overrightarrow{Y}\|F)$.
The total number of combinatorics is $\prod_{j=1}^{Q} N_j \times \binom{L}{K}$.

To calculate ever one of the function, we have Q choices in each iteration. In each iteration we consider all the set F' which is different from F at most one task, so we have at most L comparison

Exponential algorithm:

$$O(\ Q \times L \times \prod_{j=1}^{Q} N_j \times \binom{L}{K}\ )$$

Multi-threaded Caching Problem

Could we do better?

	SIZE	TIME	Complexity
Cost Model	1	$\mathbb{Z}^+$	?
Fault Model	$\mathbb{Z}^+$	1	?
General Model	$\mathbb{Z}^+$	$\mathbb{Z}^+$	NP-hard

```
for $i \leftarrow 1$ to $K$
    find the $C_{max}$ with a cache of capacity one
    delete some tasks from the input
merge the results
```

Multi-threaded Caching Problem

Could we do better?

	SIZE	TIME	Complexity
Cost Model	1	$\mathbb{Z}^+$	?
Fault Model	$\mathbb{Z}^+$	1	?
General Model	$\mathbb{Z}^+$	$\mathbb{Z}^+$	NP-hard

```
1. for $i \leftarrow 1$ to $K$
2.     find the $C_{max}$ with a cache of capacity one
3.     delete some tasks from the input
4. merge the results
```

Multi-threaded Caching Problem

Could we do better?

	SIZE	TIME	Complexity
Cost Model	1	$\mathbb{Z}^+$	?
Fault Model	$\mathbb{Z}^+$	1	?
General Model	$\mathbb{Z}^+$	$\mathbb{Z}^+$	NP-hard

```
1. for $i \leftarrow 1$ to $K$
2.     find the $C_{max}$ with a cache of capacity one
3.     delete some tasks from the input
4. merge the results
```

Multi-threaded Caching Problem

Could we do better?

	SIZE	TIME	Complexity
Cost Model	1	$\mathbb{Z}^+$	?
Fault Model	$\mathbb{Z}^+$	1	?
General Model	$\mathbb{Z}^+$	$\mathbb{Z}^+$	NP-hard

```
1. for $i \leftarrow 1$ to $K$
2.         find the $C_{max}$ with a cache of capacity one
3.         delete some tasks from the input
4. merge the results
```

Multi-threaded Caching Problem

Could we do better?

	SIZE	TIME	Complexity
Cost Model	1	$\mathbb{Z}^+$	?
Fault Model	$\mathbb{Z}^+$	1	?
General Model	$\mathbb{Z}^+$	$\mathbb{Z}^+$	NP-hard

```
for $i \leftarrow 1$ to $K$
    find the $C_{max}$ with a cache of capacity one
    delete some tasks from the input
merge the results
```

Special Case

- **input:**
  - $P_i \in \mathbb{Z}^+$, $S_i = 1 (1 \leq i \leq L)$
  - Cache of capacity ONE
  - Two chains of requests
- Observation:
  - How to save the processing time?
  - An 'edge'!

Dynamic Programming works with complexity $O(n^3)$.

Special Case

- **input:**
  - $P_i \in \mathbb{Z}^+$, $S_i = 1 (1 \leq i \leq L)$
  - Cache of capacity ONE
  - Two chains of requests
- Observation:
  - How to save the processing time?
  - An 'edge'!

Dynamic Programming works with complexity $O(n^3)$.

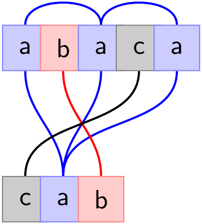
Special Case

- **input:**
  - $P_i \in \mathbb{Z}^+$, $S_i = 1 (1 \leq i \leq L)$
  - Cache of capacity ONE
  - Two chains of requests
- Observation:
  - How to save the processing time?
  - An 'edge'!

Dynamic Programming works with complexity $O(n^3)$.

Special Case

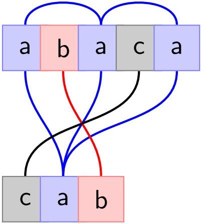
- **input:**
  - $P_i \in \mathbb{Z}^+$, $S_i = 1 (1 \leq i \leq L)$
  - Cache of capacity ONE
  - Two chains of requests
- Observation:
  - How to save the processing time?
  - An 'edge'!



Dynamic Programming works with complexity $O(n^3)$.

Special Case

- **input:**
  - $P_i \in \mathbb{Z}^+$, $S_i = 1 (1 \leq i \leq L)$
  - Cache of capacity ONE
  - Two chains of requests
- Observation:
  - How to save the processing time?
  - An 'edge'!



Dynamic Programming works with complexity $O(n^3)$.

To Be Continued . . .

- Design an approximation algorithm
- To address the complexity of Multi-threaded Caching Problem.
- Extend the number of machine
- Instead of several chains, we consider DAG
- Online Version

To Be Continued . . .

- Design an approximation algorithm
- To address the complexity of Multi-threaded Caching Problem.
- Extend the number of machine
- Instead of several chains, we consider DAG
- Online Version

To Be Continued . . .

- Design an approximation algorithm
- To address the complexity of Multi-threaded Caching Problem.
- Extend the number of machine
- Instead of several chains, we consider DAG
- Online Version

To Be Continued . . .

- Design an approximation algorithm
- To address the complexity of Multi-threaded Caching Problem.
- Extend the number of machine
- Instead of several chains, we consider DAG
- Online Version

To Be Continued . . .

- Design an approximation algorithm
- To address the complexity of Multi-threaded Caching Problem.
- Extend the number of machine
- Instead of several chains, we consider DAG
- Online Version

Merci !