Introduction to Sporadic Tasks and Real-Time Synchronization

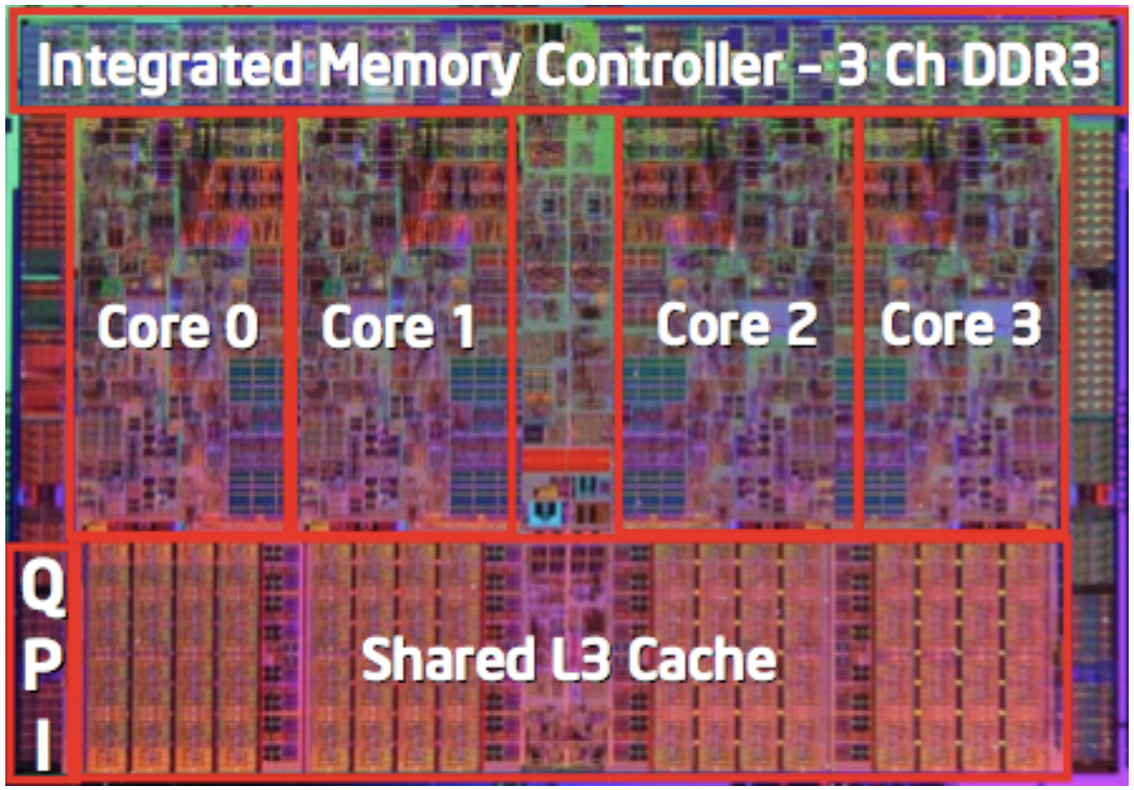
– 9/9/09 –

Björn Brandenburg

Outline

1. **Introduction**
2. Real-Time Scheduling
3. Real-Time Synchronization
4. ~~Research Agenda~~
5. ~~Summary~~

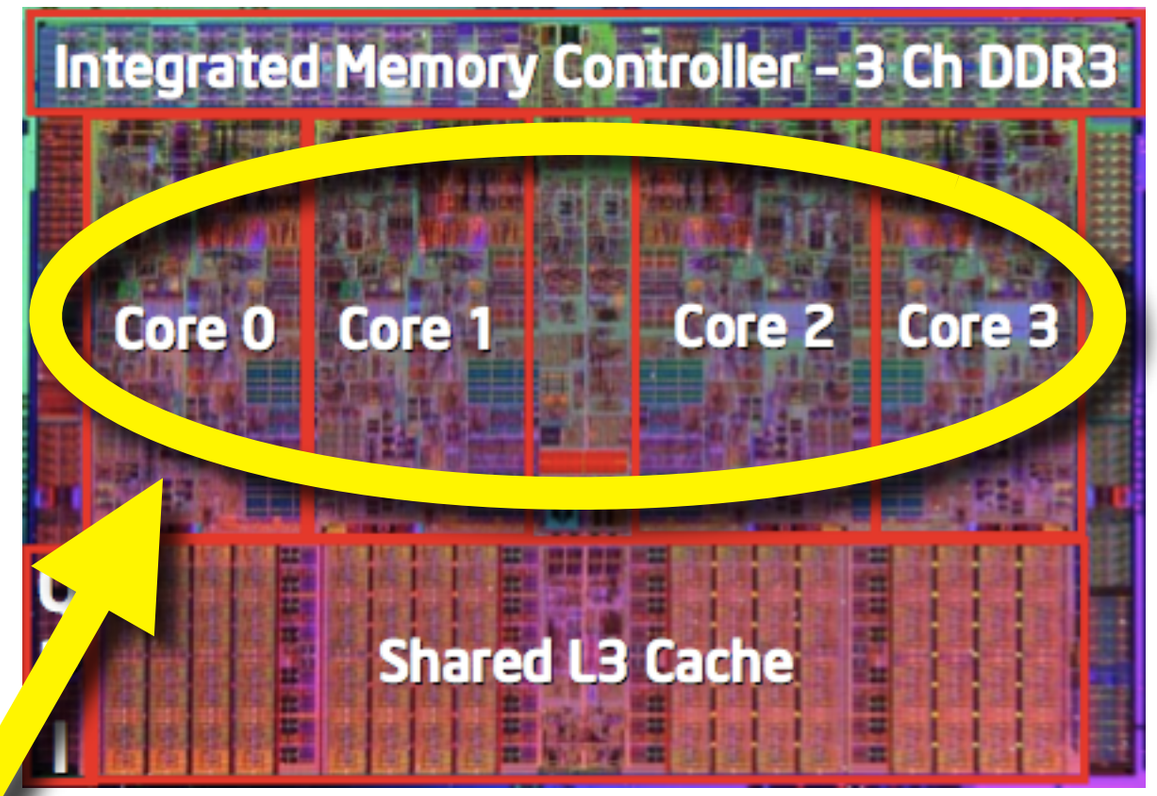
Multicore



Intel Nehalem

Image credit: www.devicedaily.com

Multicore

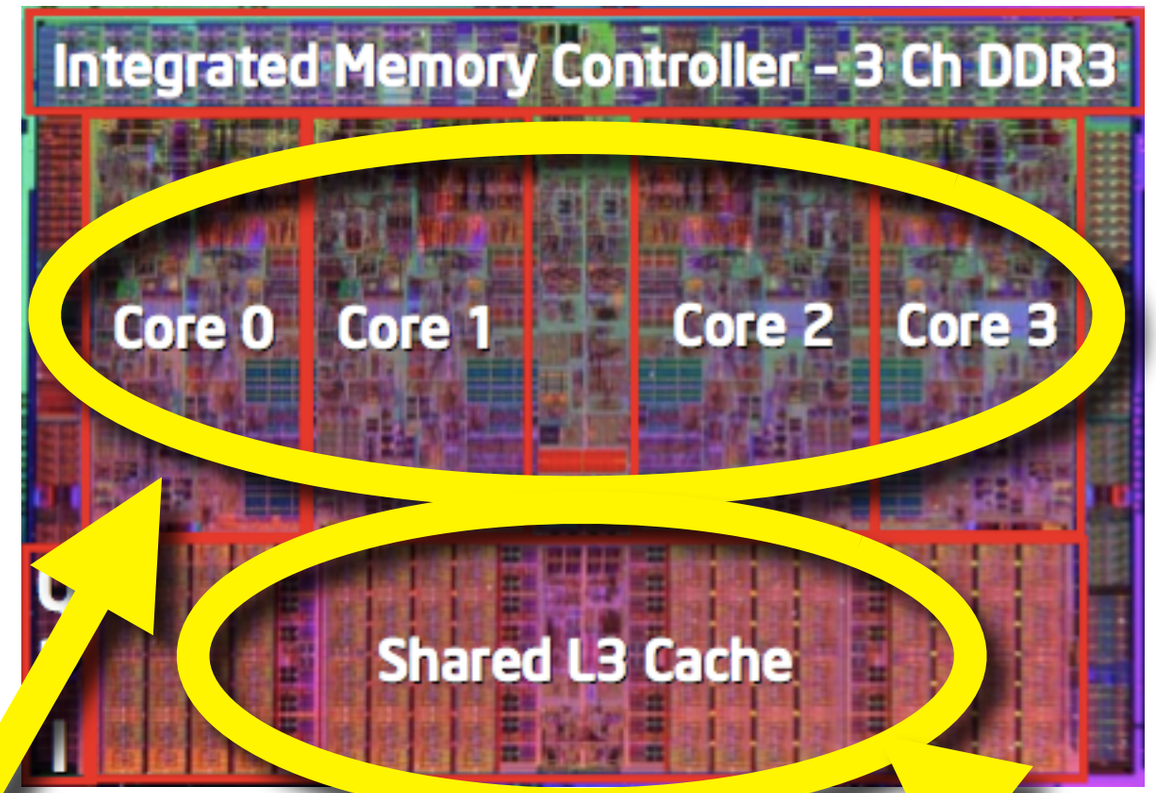


Intel Nehalem

Multiprocessors are now the common case.

Image credit: www.devicedaily.com

Multicore



Intel Nehalem

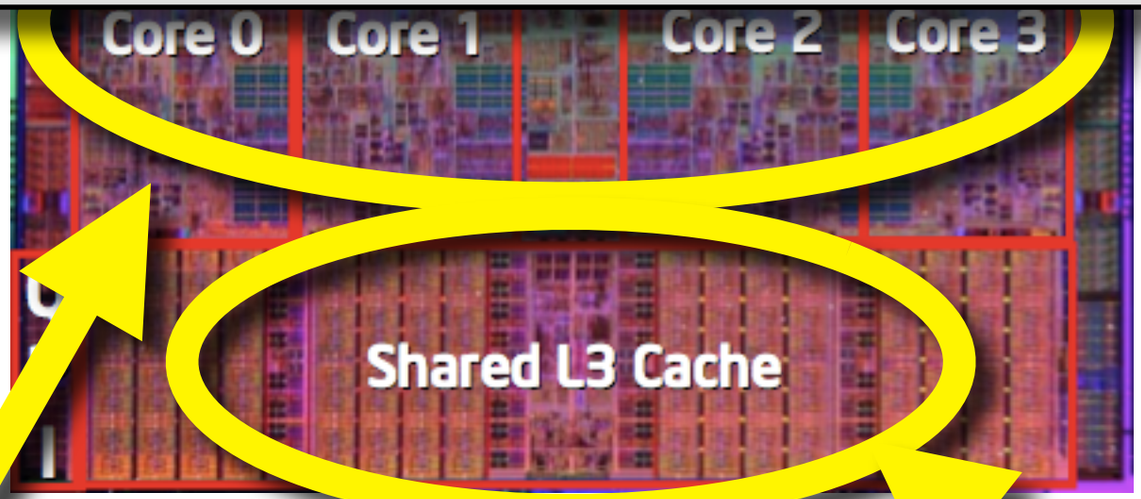
Multiprocessors are now the common case.

Some caches are shared between processors.

Image credit: www.devicedaily.com

Multicore

processor
=
schedulable hardware context exposed to the OS



Intel Nehalem

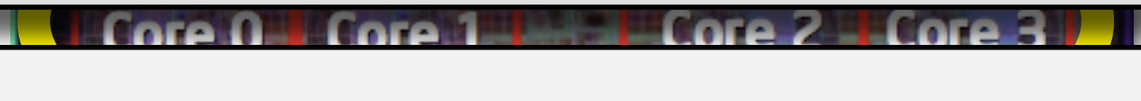
Multiprocessors are now the common case.

Some caches are shared between processors.

Image credit: www.devicedaily.com

Multicore

processor
=
schedulable hardware context exposed to the OS



#processors
=
#chips × #cores per chip × #threads per core

Multiprocessors are now the common case.

Some caches are shared between processors.

Image credit: www.devicedaily.com

Why use multicore computers for real-time systems?

1. To save money.

- may save on power, cooling, weight, wiring, etc.
- multicore: good performance/price ratio

Why use multicore computers for real-time systems?

1. To save money.

- may save on power, cooling, weight, wiring, etc.
- multicore: good performance/price ratio

2. High performance needed.

- e.g., HDTV, other high-quality multimedia apps
- real-time business transaction processing...

One example: AZUL Systems, Inc.

AZUL builds special-purpose transaction processing appliances. They consist of up to **864 cores**.

"Consistent, Fast Response Times

When critical business applications pause, companies lose money. When it comes to fulfilling on-line purchases, executing stock trades at the real time price, acting on price fluctuations or approving loan applications, **completing only 85 percent of the requests in time is a failure**."

Source: http://www.azulsystems.com/products/compute_appliance.htm?p=p

One example: AZ

Predictability

AZUL builds special-purpose t
appliances. They consist of up to **864 cores**.

"Consistent, Fast Response Times

When critical business applications pause, companies lose money. When it comes to fulfilling on-line purchases, executing stock trades at the real time price, acting on price fluctuations or approving loan applications, **completing only 85 percent of the requests in time is a failure**."

Source: http://www.azulsystems.com/products/compute_appliance.htm?p=p

One example: AZ

Predictability

AZUL builds special-purpose t
appliances. They consist f up to **864 cores**.

"Consistent Fast Response Times

When critical siness applications pause, companies lose money. When it comes to chases, executing stock trades at acting on price fluctuations or ications, **completing only 85 percent of the requests in time is a failure**."

Low latency

Source: http://www.azulsystems.com/products/compute_appliance.htm?p=p

One example: AZUL Systems, Inc.

AZ
app

Correctness depends on **temporal correctness**

=

A big multicore real-time system!

When critical busine s applications pause, companies lose mon y. When it comes to fulfilling on-line purchases, executing stock trades at the real time price, acting on price fluctuations o approving loan applications, **completing only 85 percent of the requests in time is a failure**."

Source: http://www.azulsystems.com/products/compute_appliance.htm?p=p

"Amazon found every ***100ms of latency*** *cost them* ***1% in sales***.

Google found an extra ***.5 seconds in search page generation time*** ***dropped traffic by 20%***.

A broker could ***lose $4 million*** *in revenues* ***per millisecond*** *if their electronic trading platform is* ***5 milliseconds*** *behind the competition."*

Source: http://highscalability.com/latency-everywhere-and-it-costs-you-sales-how-crush-it and http://www.tabbgroup.com/PublicationDetail.aspx?PublicationID=346

"If a broker is
100 milliseconds slower than the fastest broker,
it may as well shut down
its [trading system] and become a floor broker."

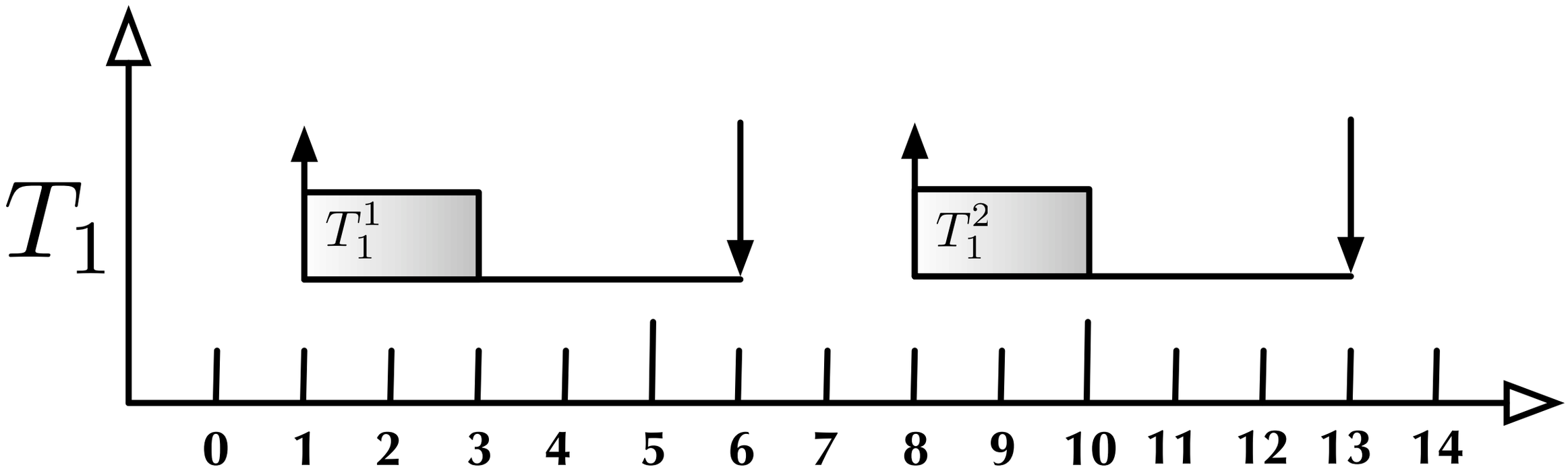
A broker could ***lose $4 million*** *in revenues* ***per millisecond*** *if their electronic trading platform is* ***5 milliseconds*** *behind the competition."*

Source: http://highscalability.com/latency-everywhere-and-it-costs-you-sales-how-crush-it and http://www.tabbgroup.com/PublicationDetail.aspx?PublicationID=346

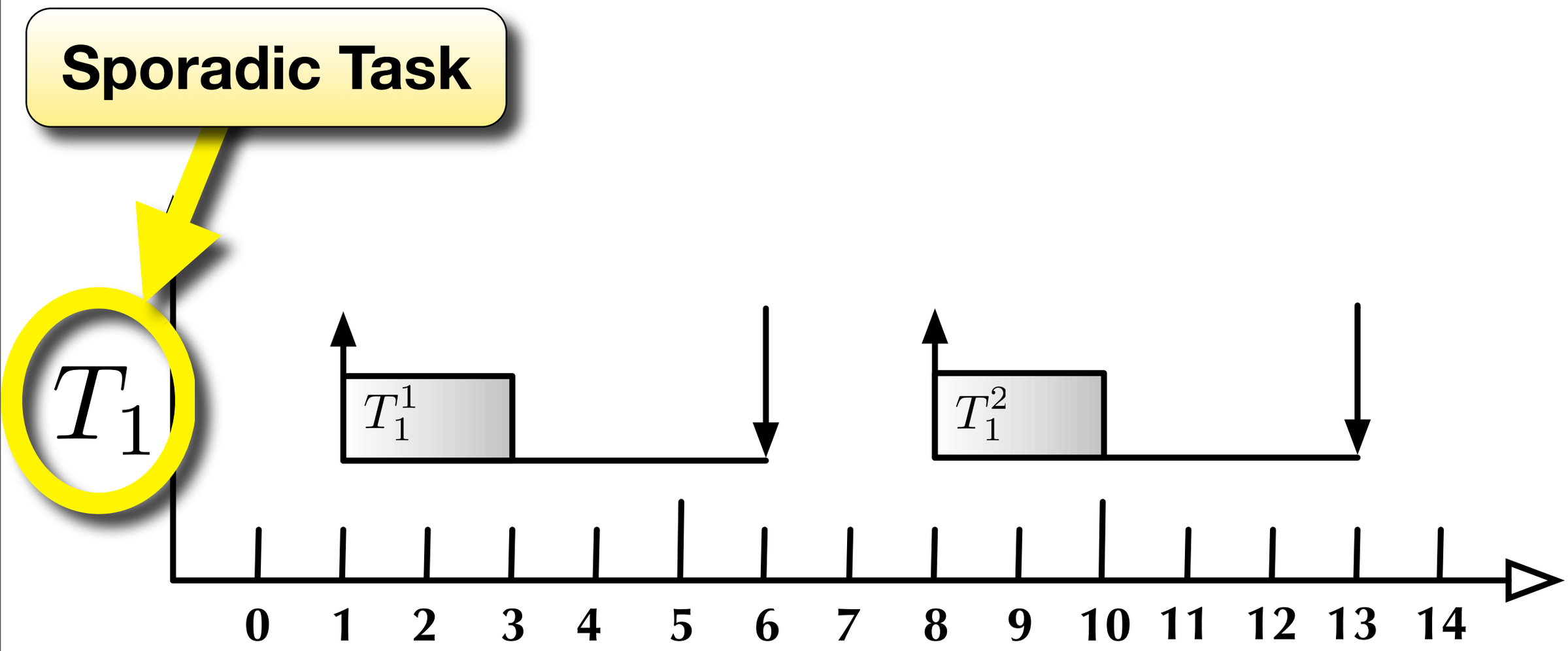
Outline

1. Introduction
2. **Real-Time Scheduling**
3. Real-Time Synchronization
4. ~~Research Agenda~~
5. ~~Summary~~

Sporadic Task Model



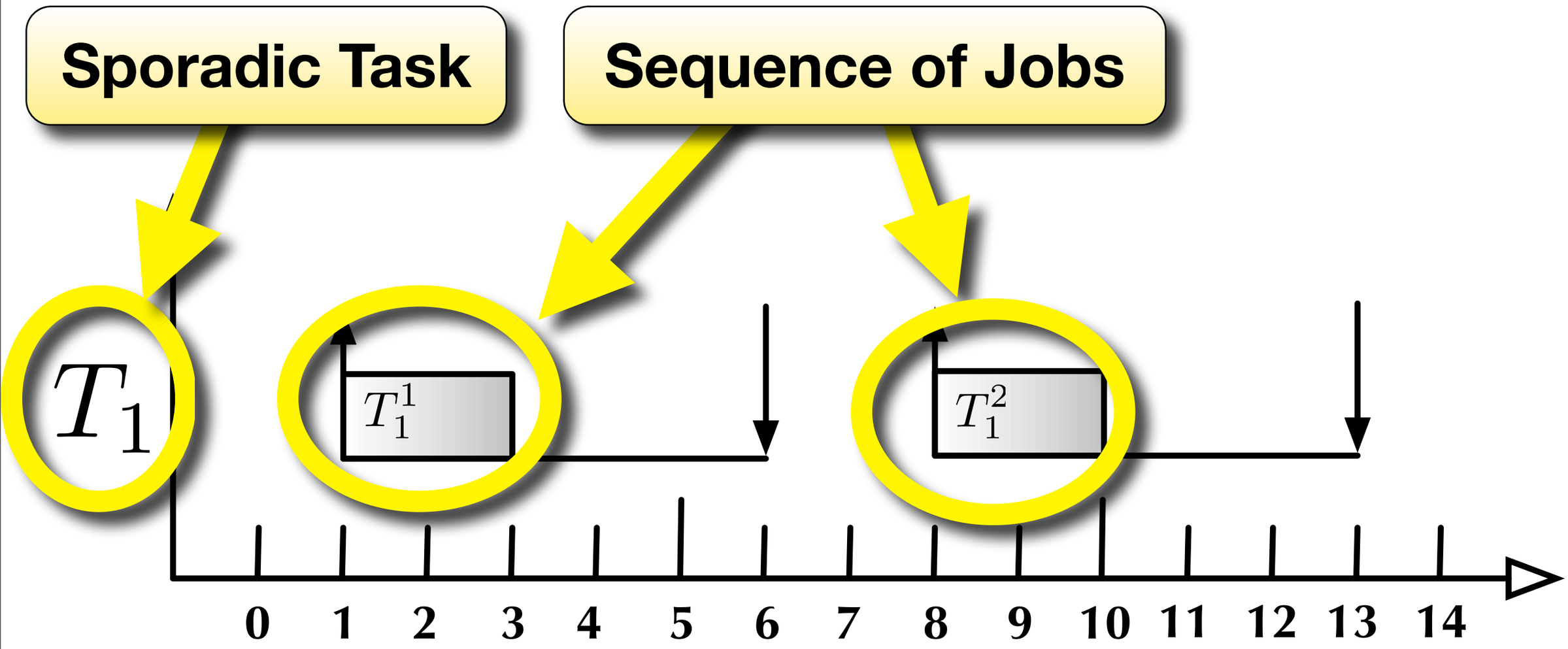
Sporadic Task Model



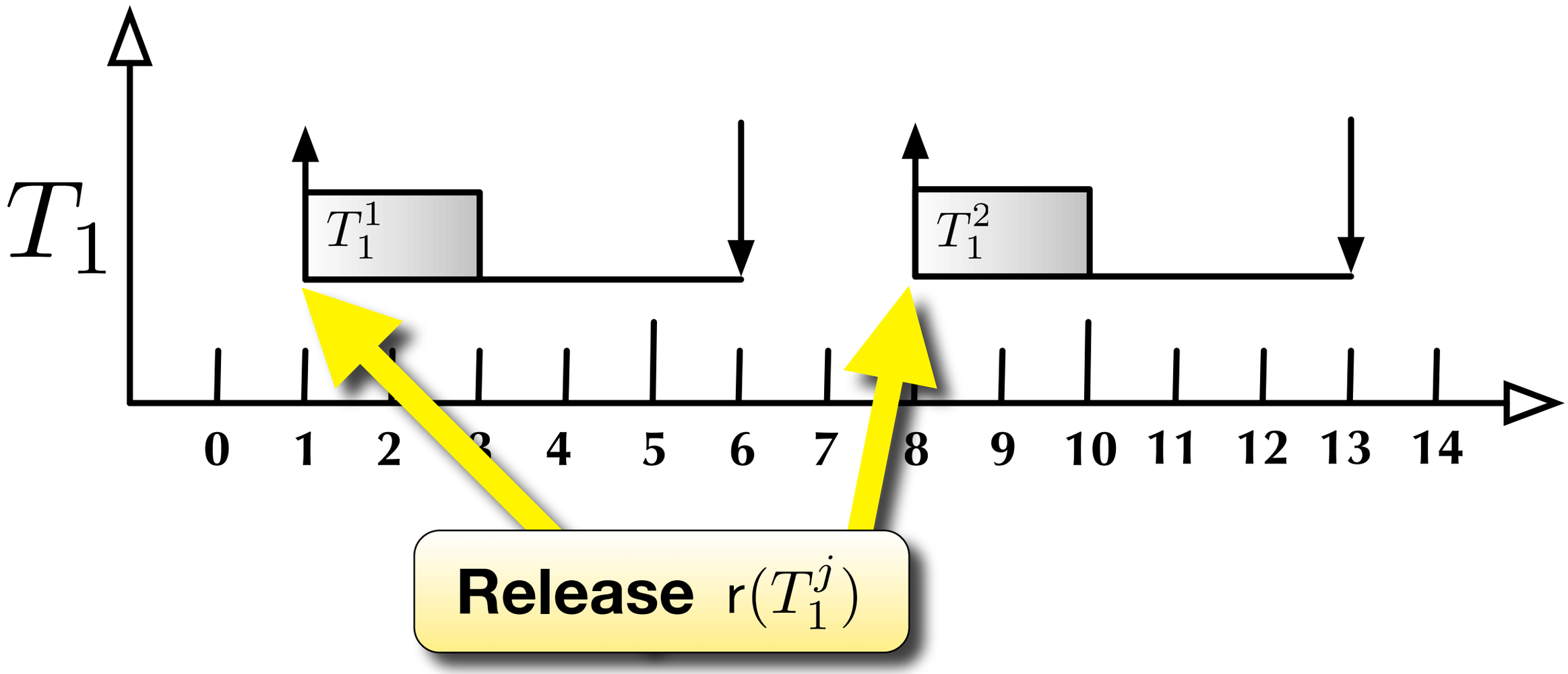
Sporadic Task Model

Sporadic Task

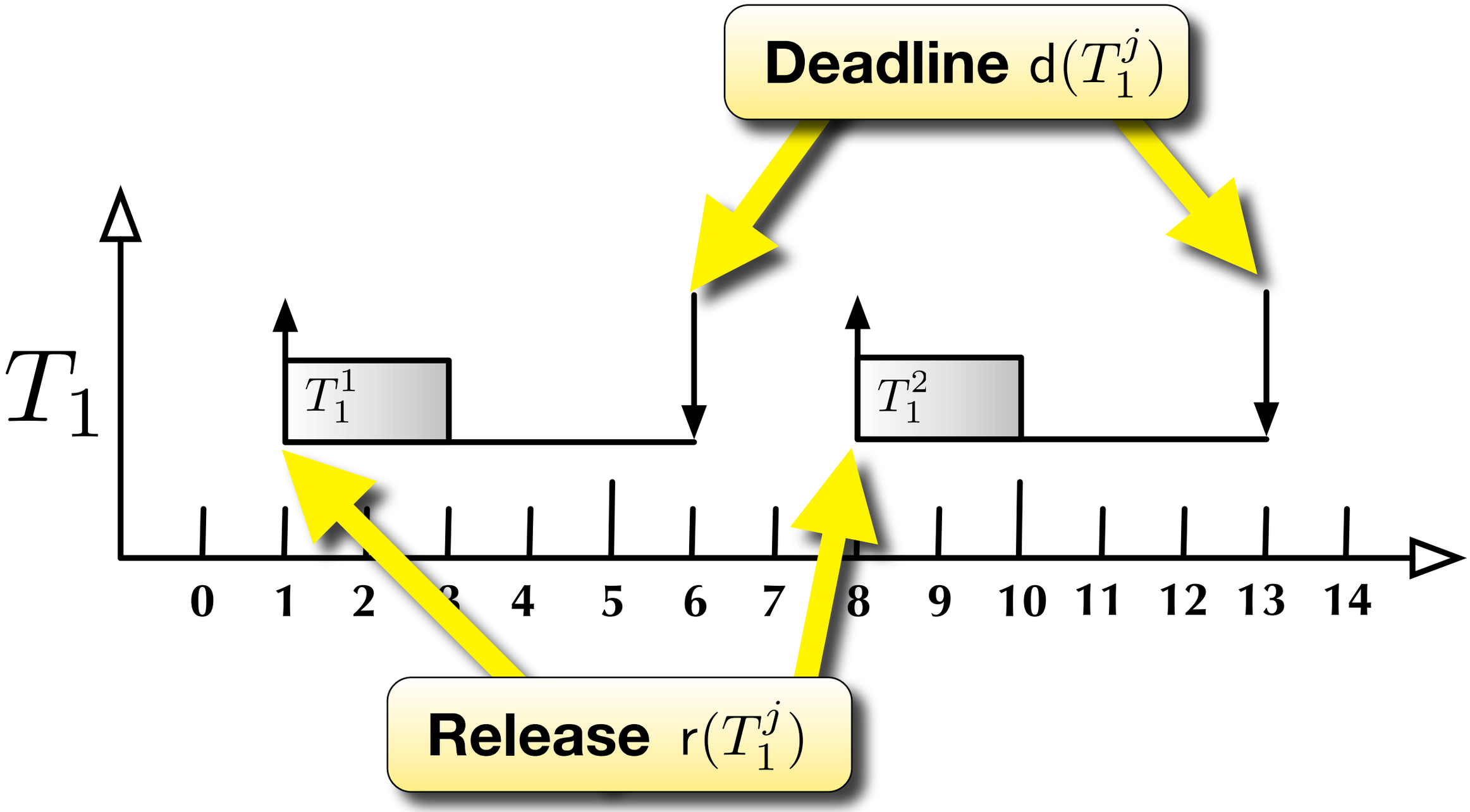
Sequence of Jobs



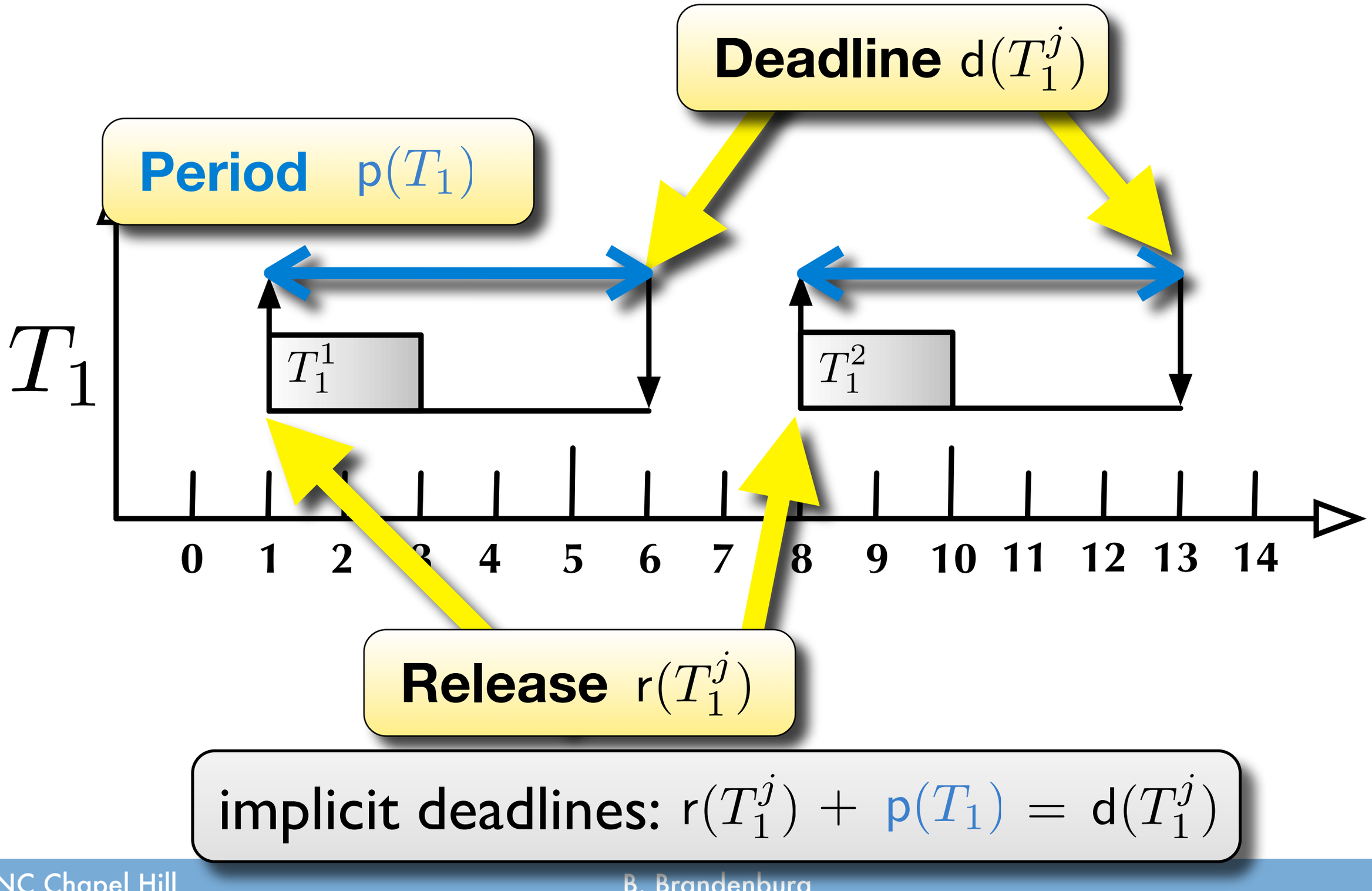
Sporadic Task Model



Sporadic Task Model

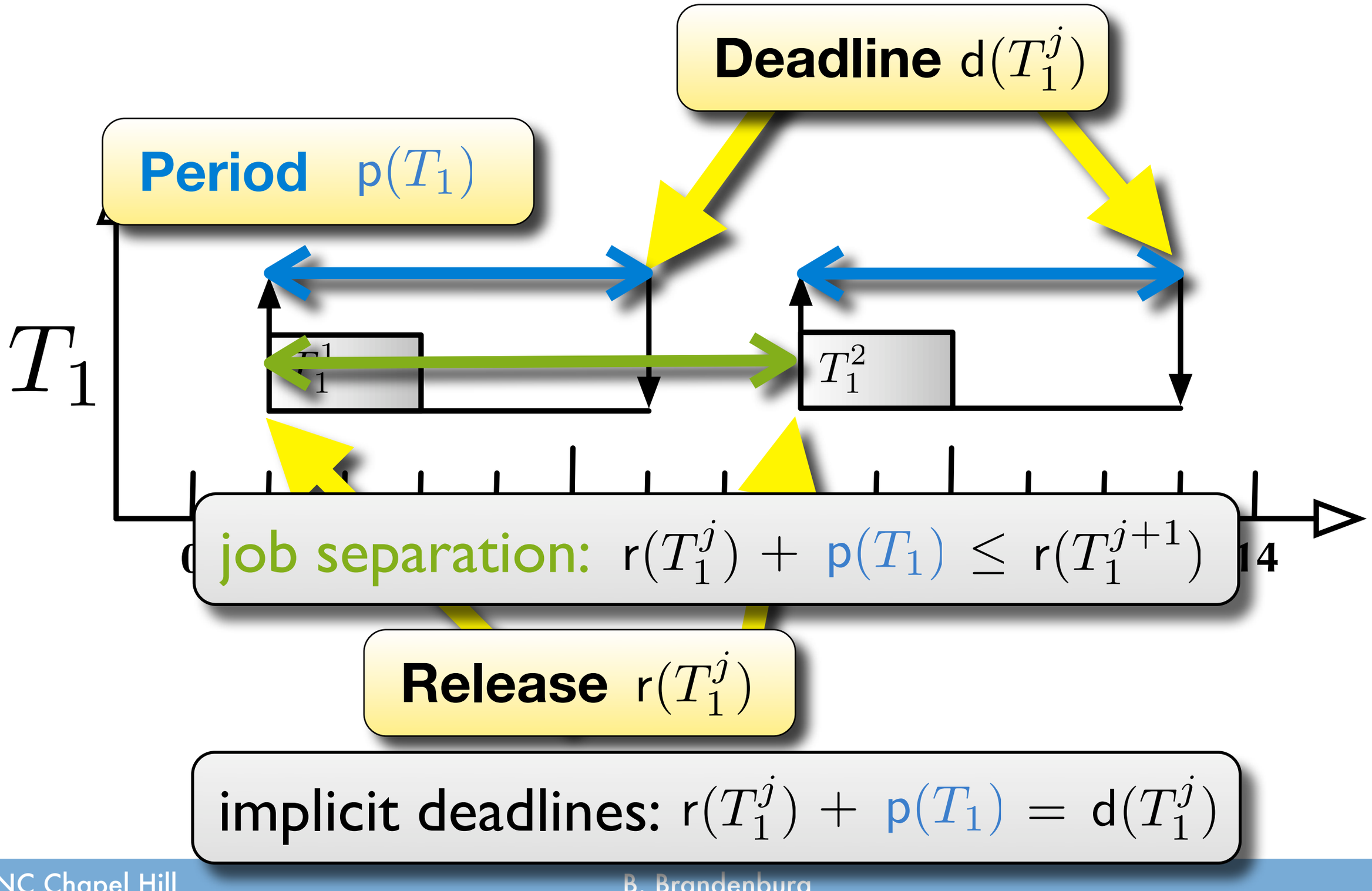


Sporadic Task Model

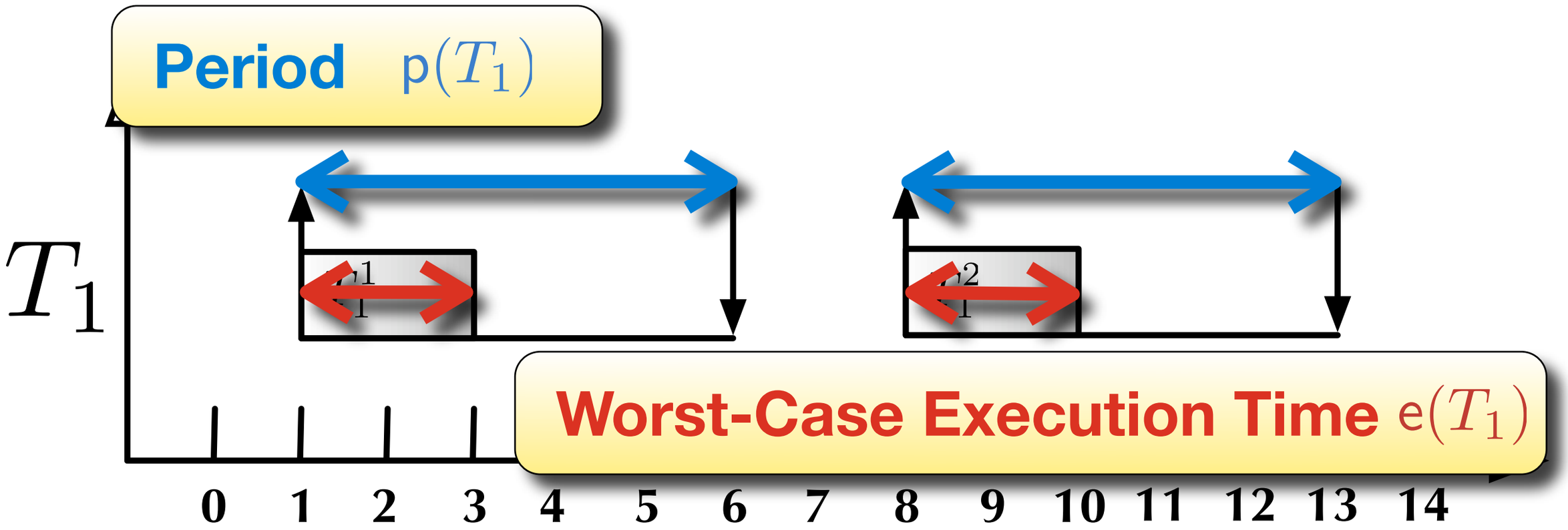


implicit deadlines: $\mathrm{r}(T_1^j) + \mathrm{p}(T_1) = \mathrm{d}(T_1^j)$

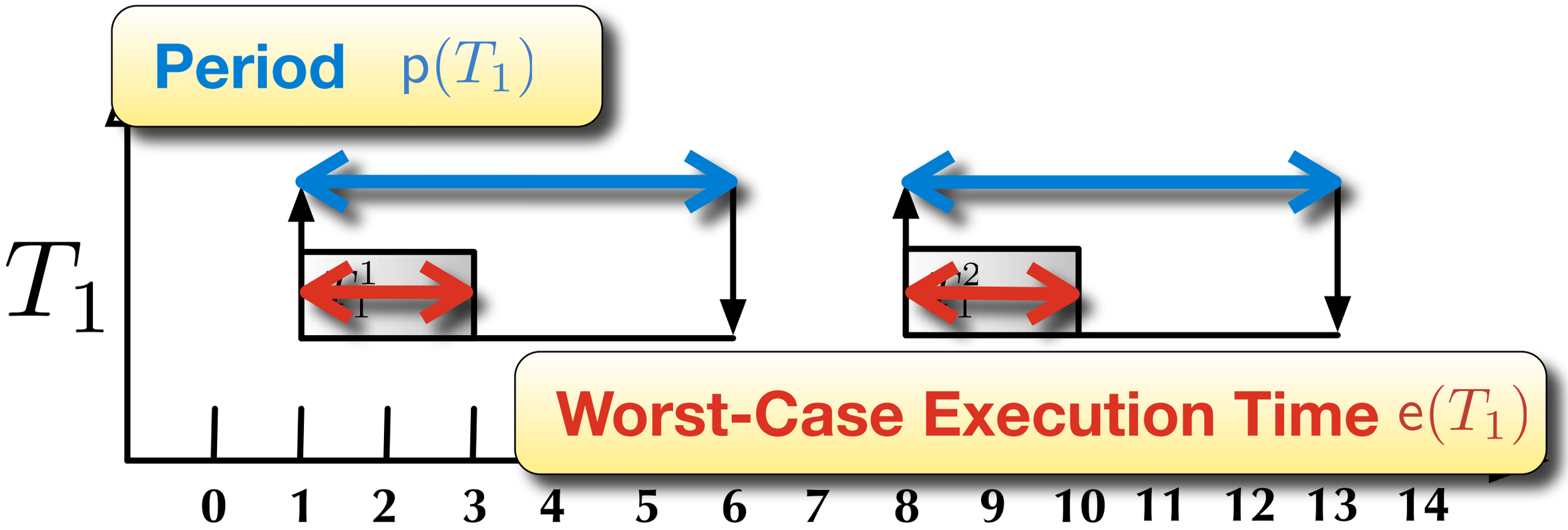
Sporadic Task Model



Sporadic Task Model



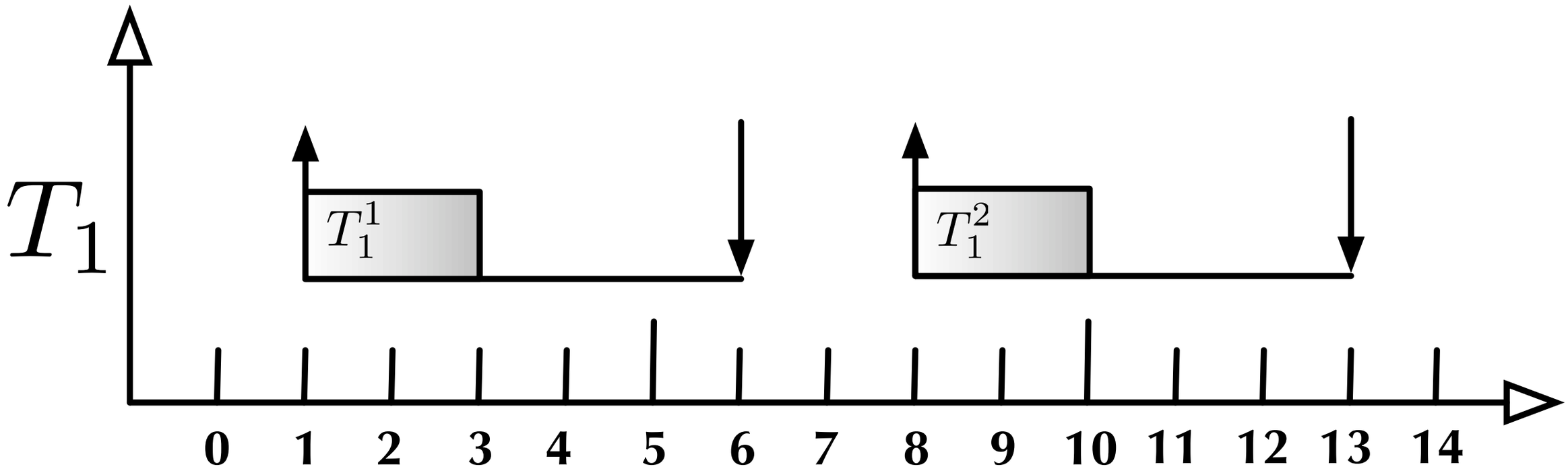
Sporadic Task Model



utilization: $\dfrac{\mathsf{e}(T_1)}{\mathsf{p}(T_1)} = \mathsf{u}(T_1)$

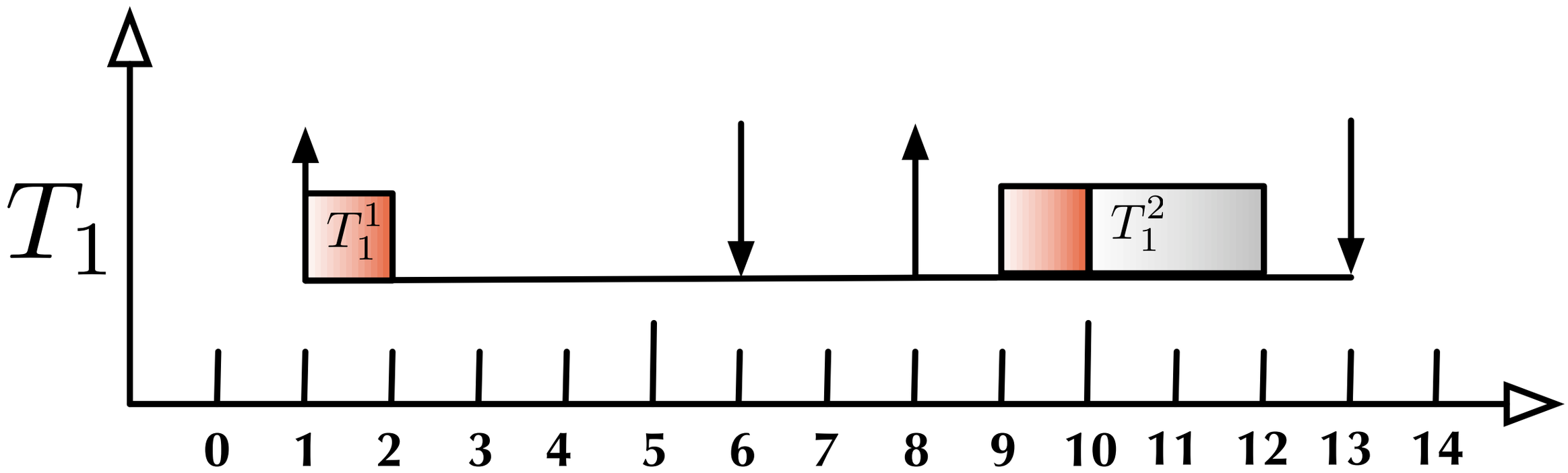
Tardiness

What happens when a job does not complete on time?



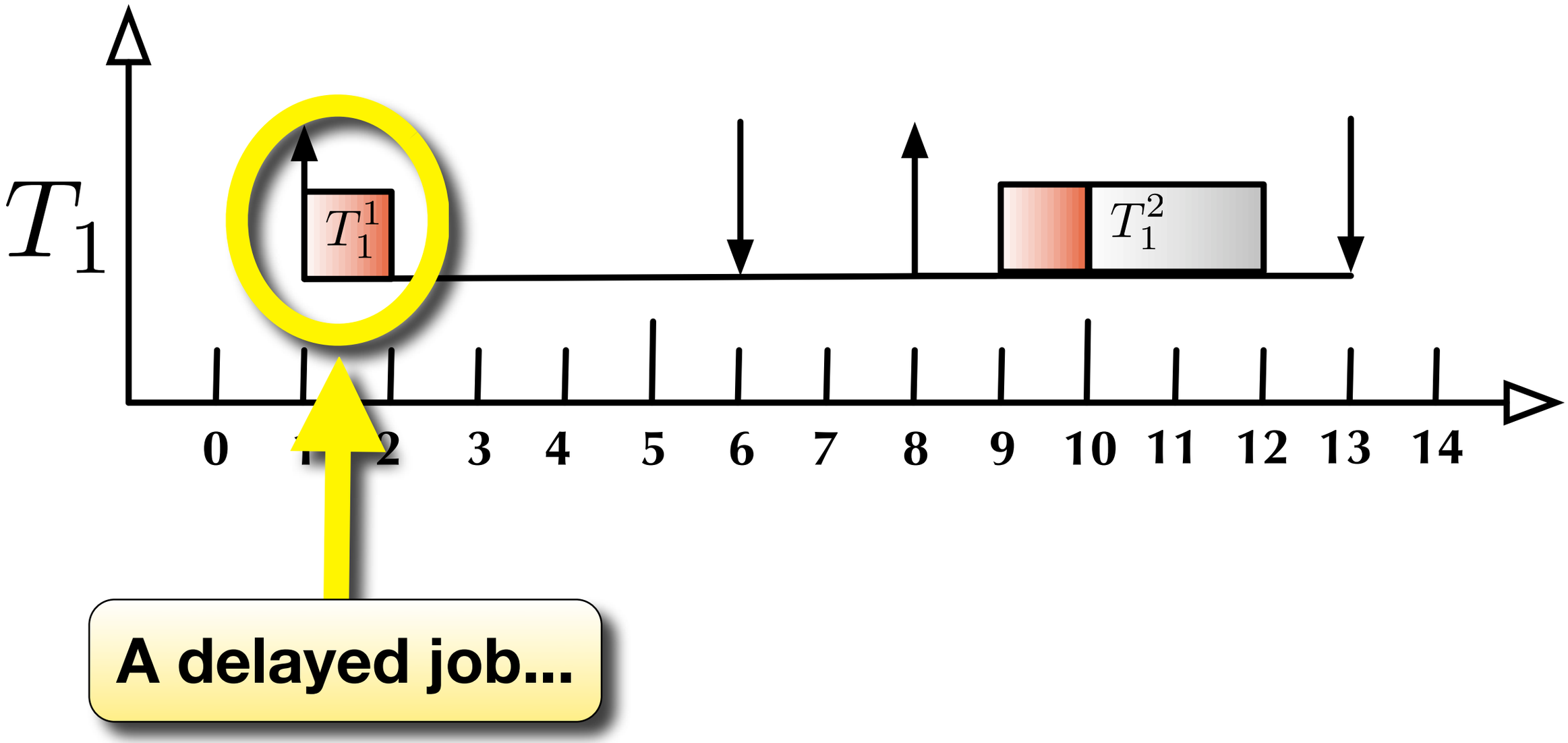
Tardiness

What happens when a job does not complete on time?



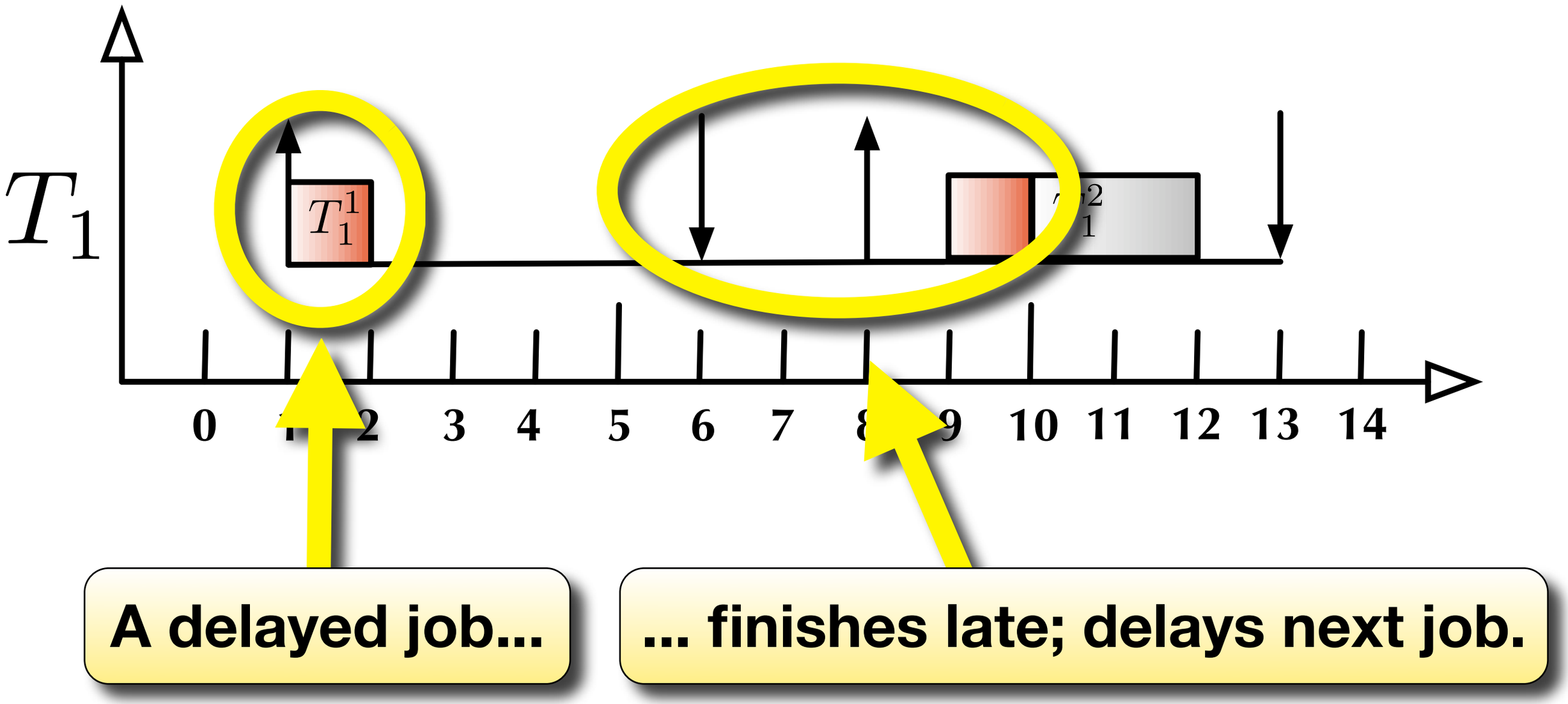
Tardiness

What happens when a job does not complete on time?



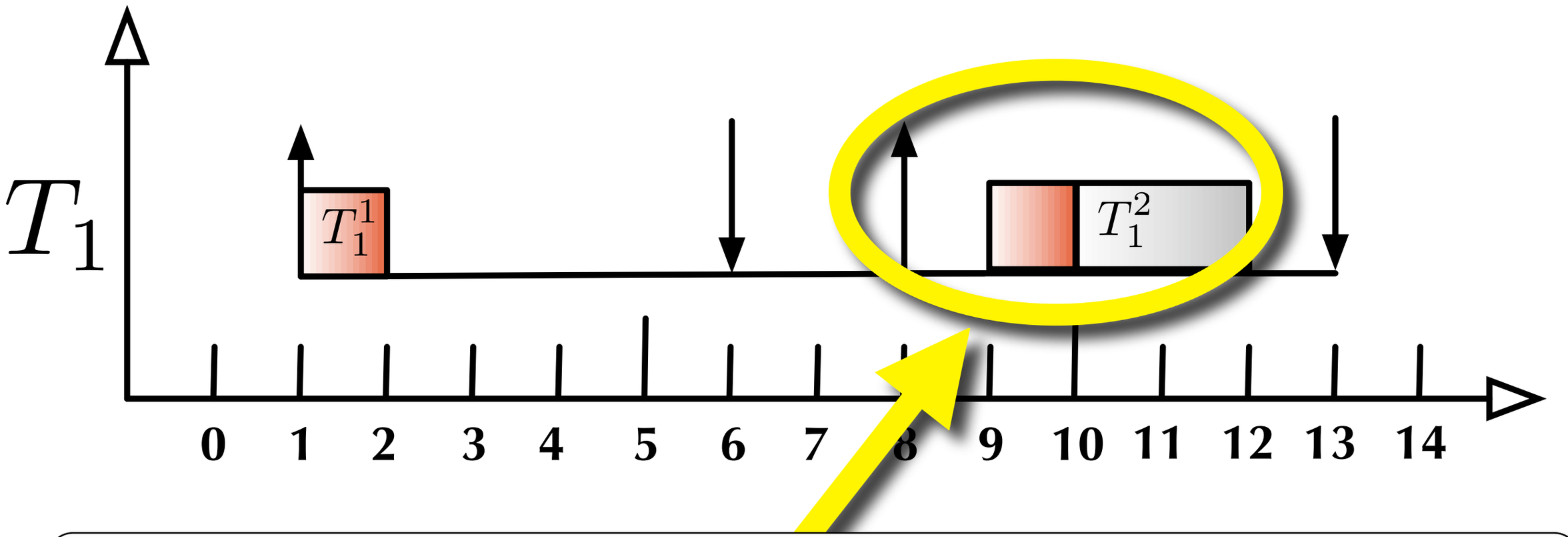
Tardiness

What happens when a job does not complete on time?



Tardiness

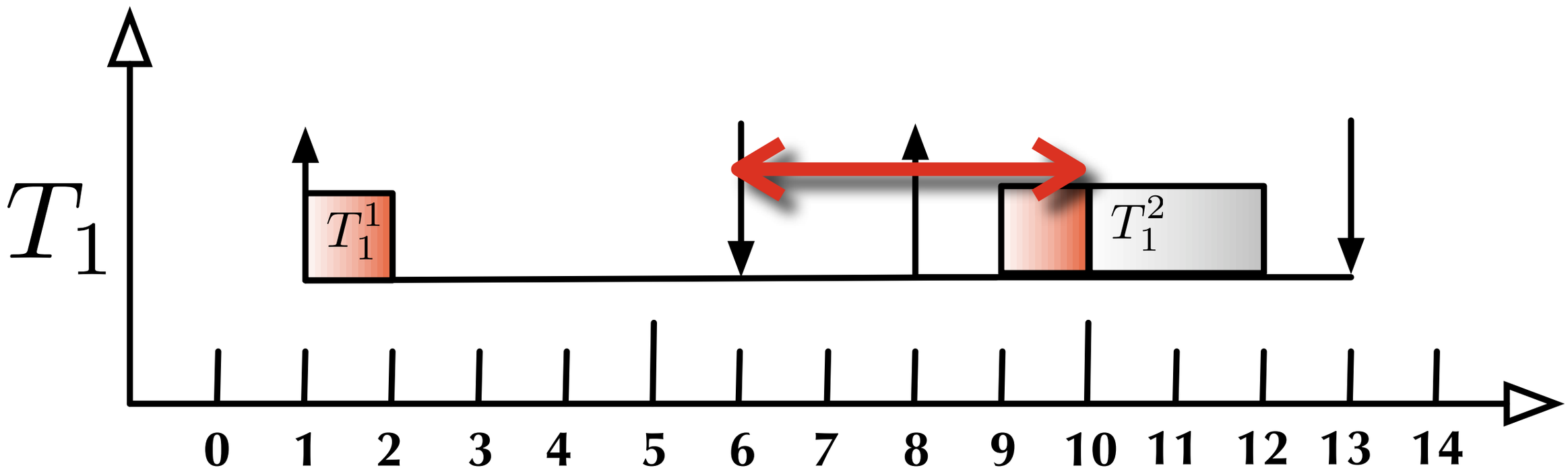
What happens when a job does not complete on time?



Tasks are sequential:
next job cannot be scheduled until prior job completes.

Tardiness

What happens when a job does not complete on time?



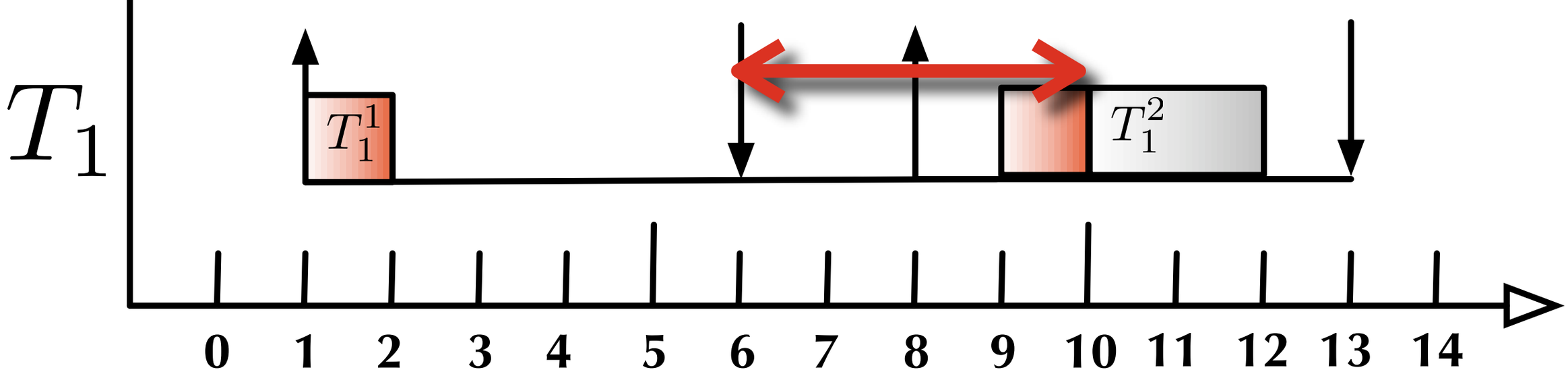
Tardiness

hard real-time (HRT)

all jobs finish on time
= zero tardiness

diness

does not complete on time?



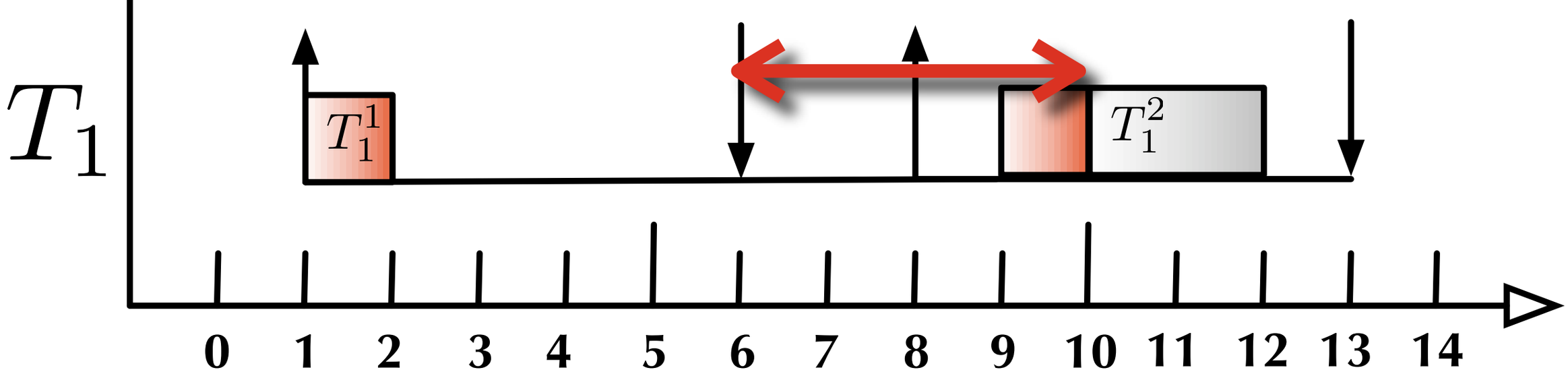
Tardiness

hard real-time (HRT)

all jobs finish on time
= zero tardiness

soft real-time (SRT)

maximum tardiness is bounded



Tardiness

Uniprocessor Scheduling

Uniprocessor Scheduling

Earliest Deadline First (EDF)

=

Execute pending jobs

in order of non-decreasing deadline;

break ties arbitrarily.

Uniprocessor Scheduling

Earliest Deadline First (EDF)

=

Execute pending jobs
in order of non-decreasing deadline;
break ties arbitrarily.

Static Priority (SP)

=

Assign unique priorities to tasks;
execute pending jobs
in order of decreasing task priority.

Uniprocessor Scheduling

Earliest Deadline First (EDF)

EDF is optimal:

all deadlines met
if system not over-utilized

Static Priority (SP)

=

Assign unique priorities to tasks;
execute pending jobs
***in order of decreasing task priority*.**

Uniprocessor Scheduling

Earliest Deadline First (EDF)

EDF is optimal:

all deadlines met
if system not over-utilized

Static Priority (SP)

SP is not optimal:

meeting all deadlines may require cap on utilization (idle time)

Multiprocessor Scheduling

Multiprocessor Scheduling

Partitioning

=

use uniprocessor algorithm
on each processor

Multiprocessor Scheduling

Partitioning

=

use uniprocessor algorithm
on each processor

Global

=

one global run queue;
served by all processors

Multiprocessor Scheduling

Partitioning

=

use uniprocessor algorithm
on each processor

Global

=

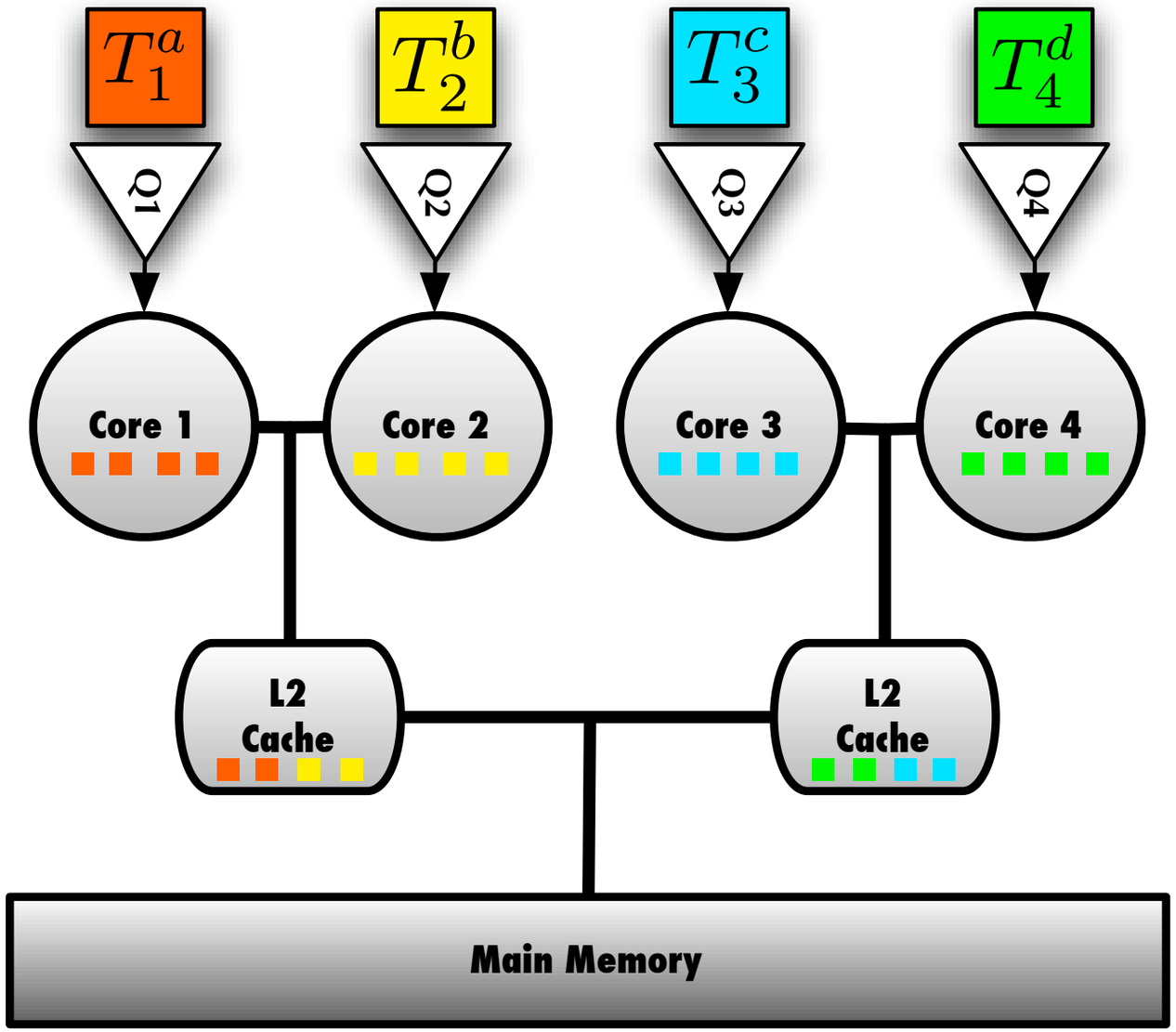
one global run queue;
served by all processors

Clustered

=

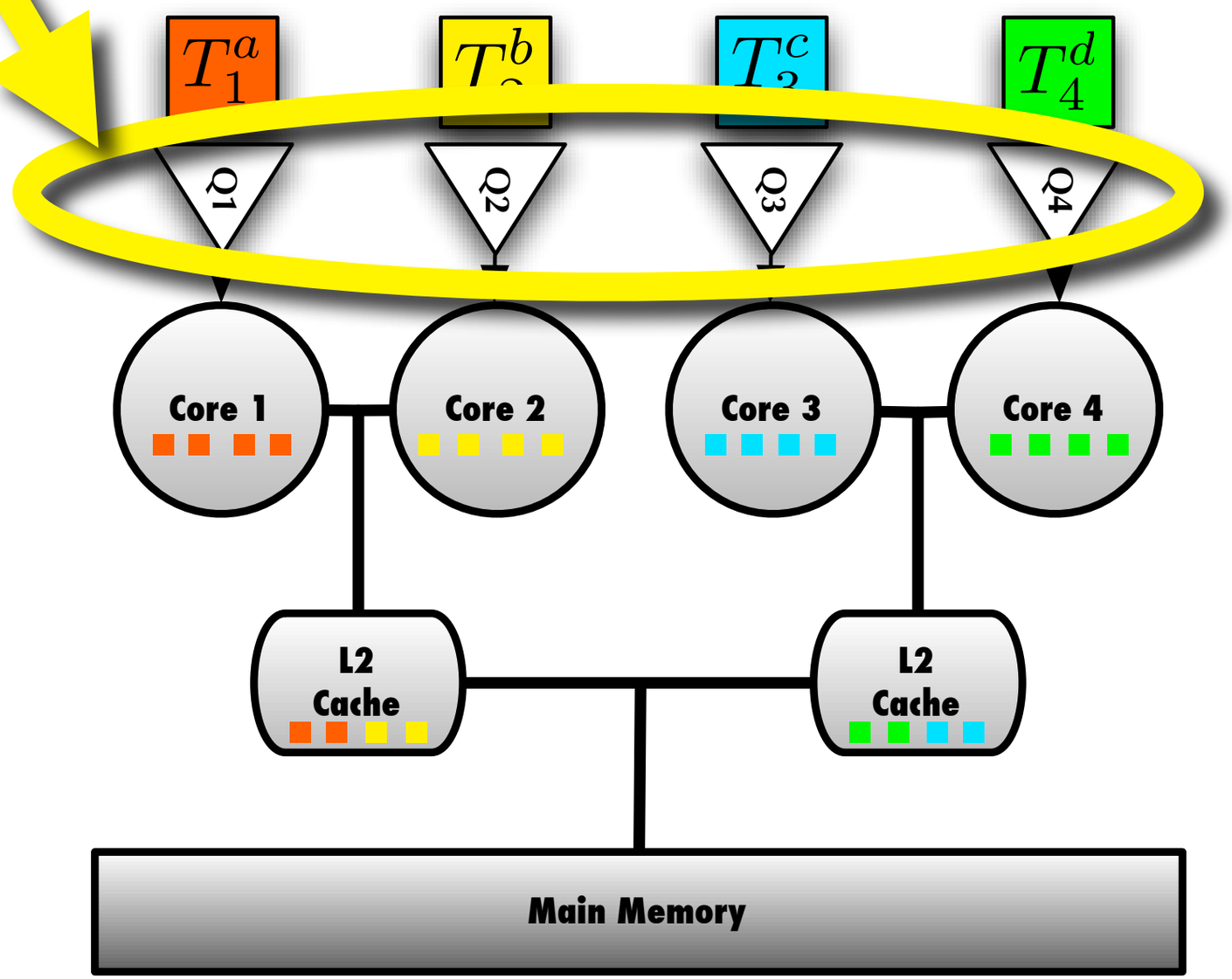
globally schedule
clusters of processors

Partitioning



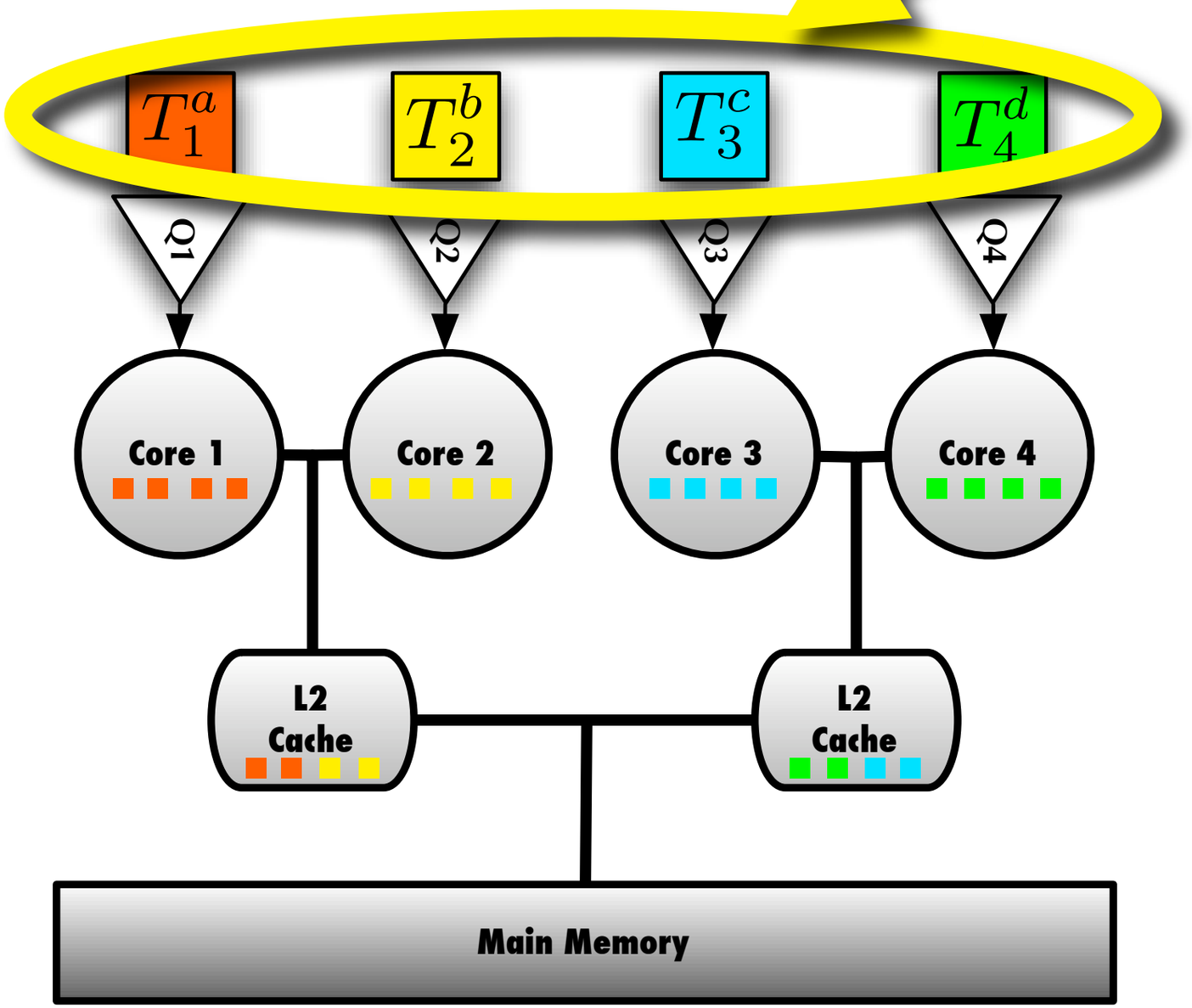
Partitioning

One run queue per processor.



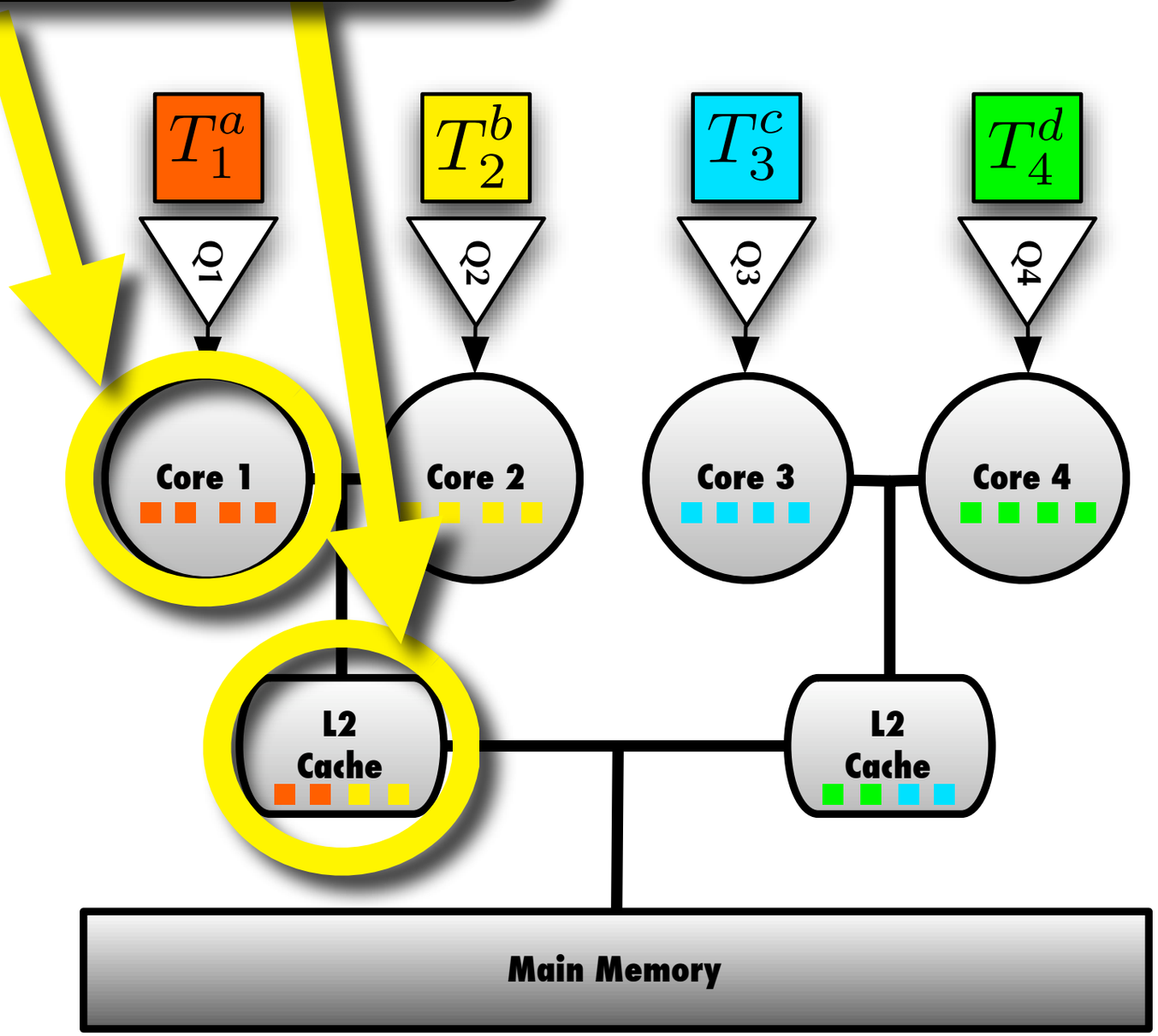
Partitioning

Tasks are assigned statically to processors.



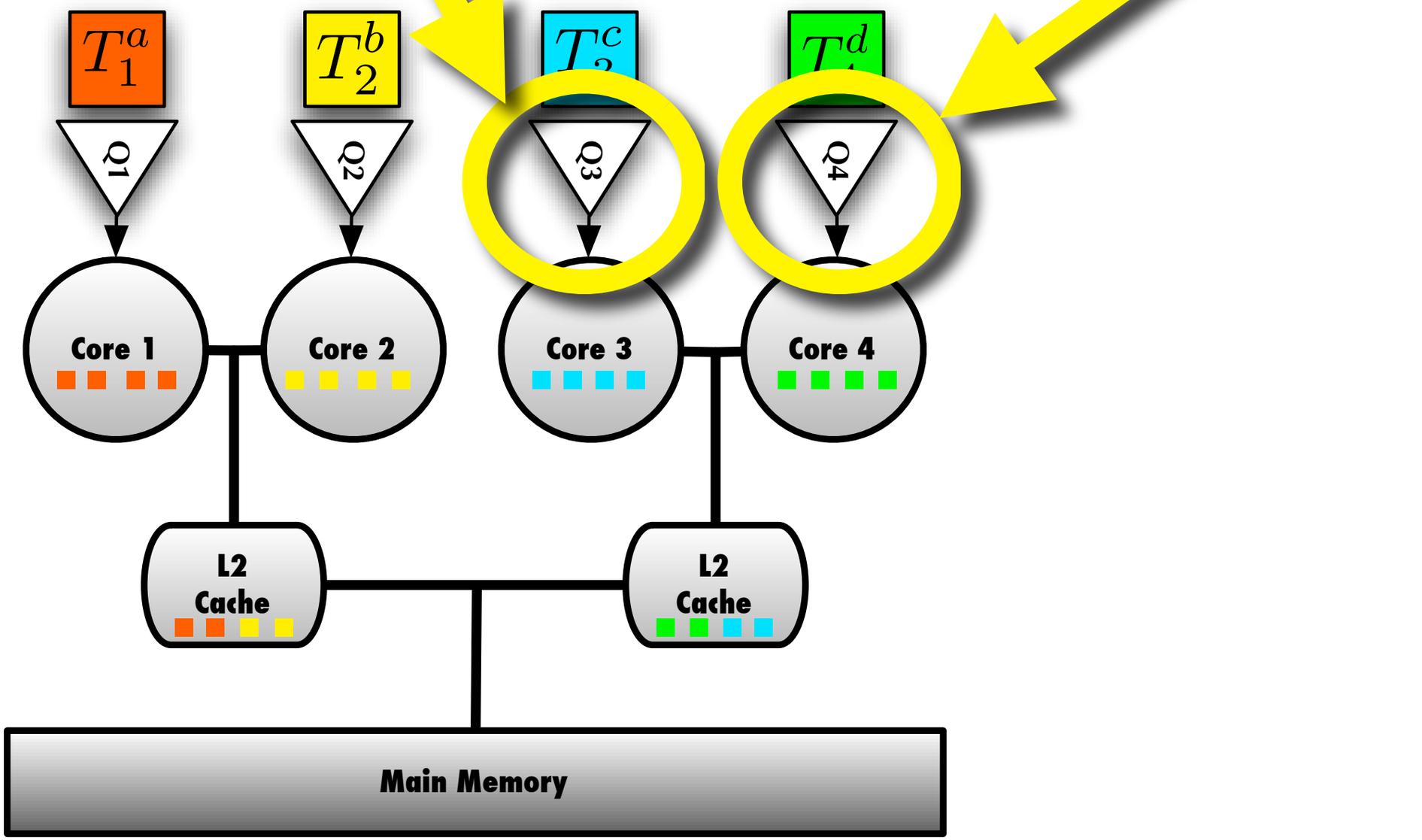
Partitioning

Good cache affinity.



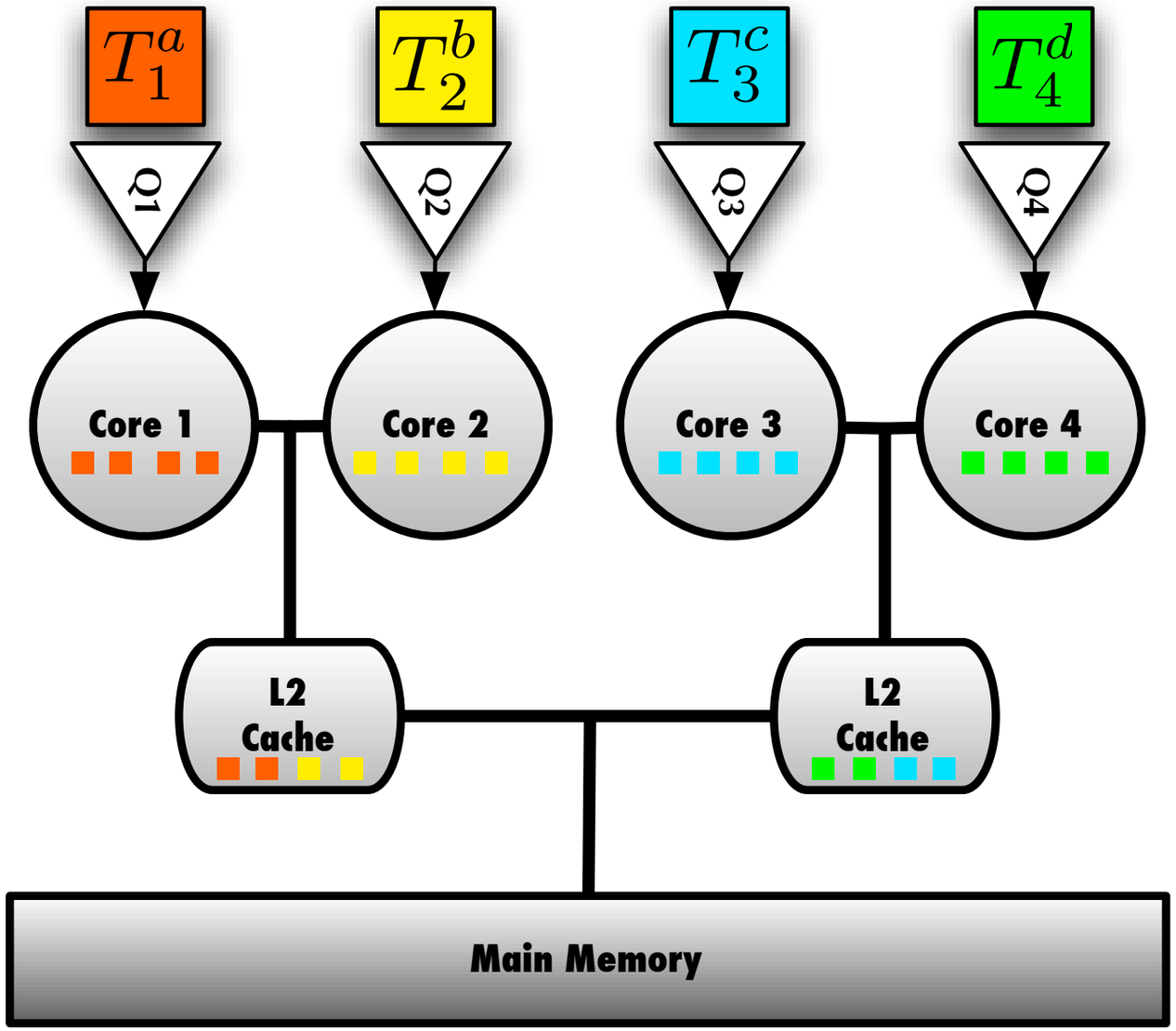
Partitioning

Low queue contention:
processors access mostly local queues.



Partitioning

But: partitioning requires a bin-packing problem to be solved...



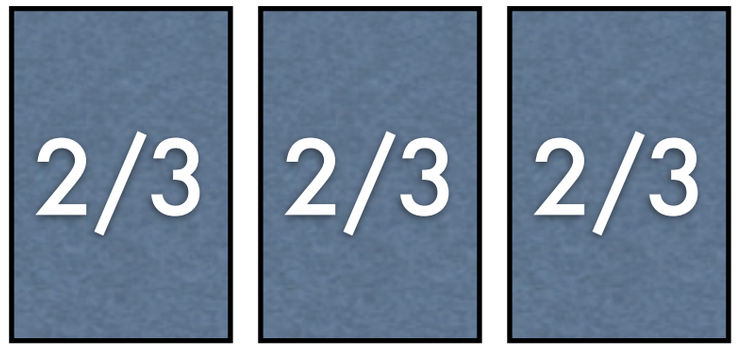
Partitioning

Example: three identical tasks

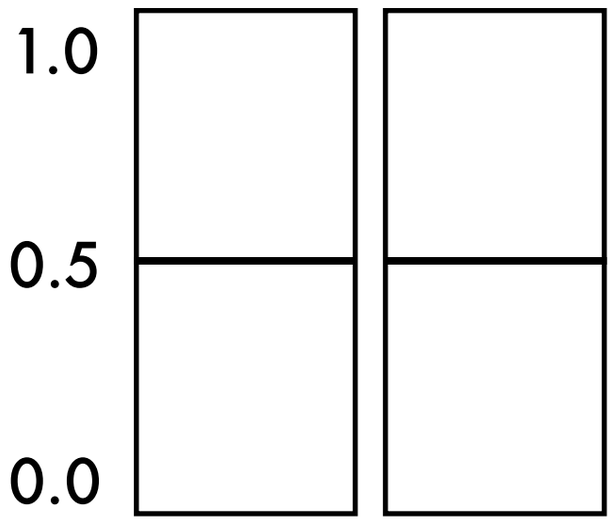
period = 3

wcet = 2

util. = 2/3



two unit processors

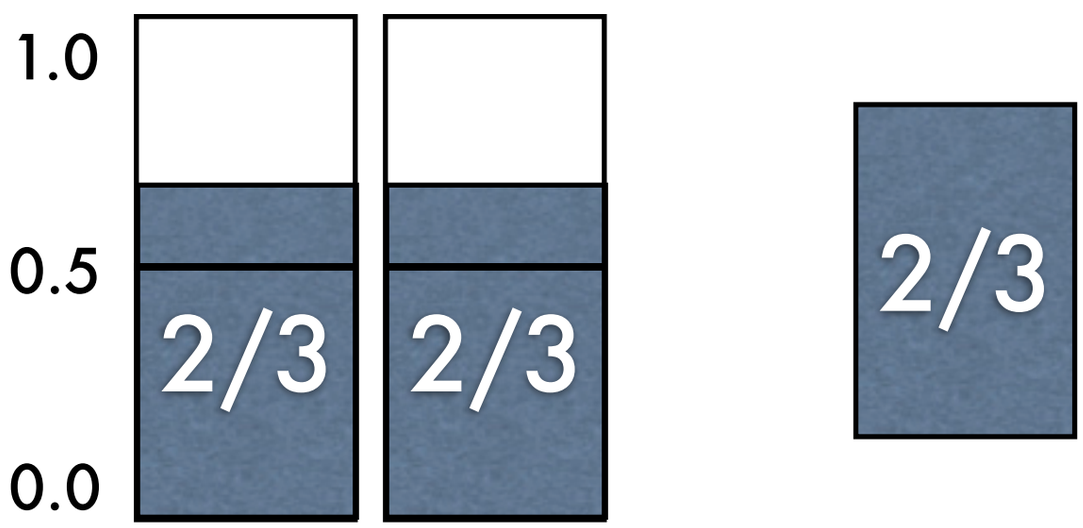


Partitioning

Bin packing

Even though there is sufficient total capacity, the last task cannot be placed.

two unit processors

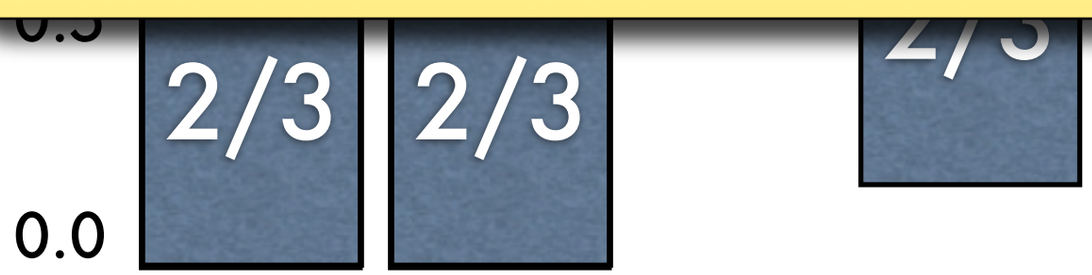


Partitioning

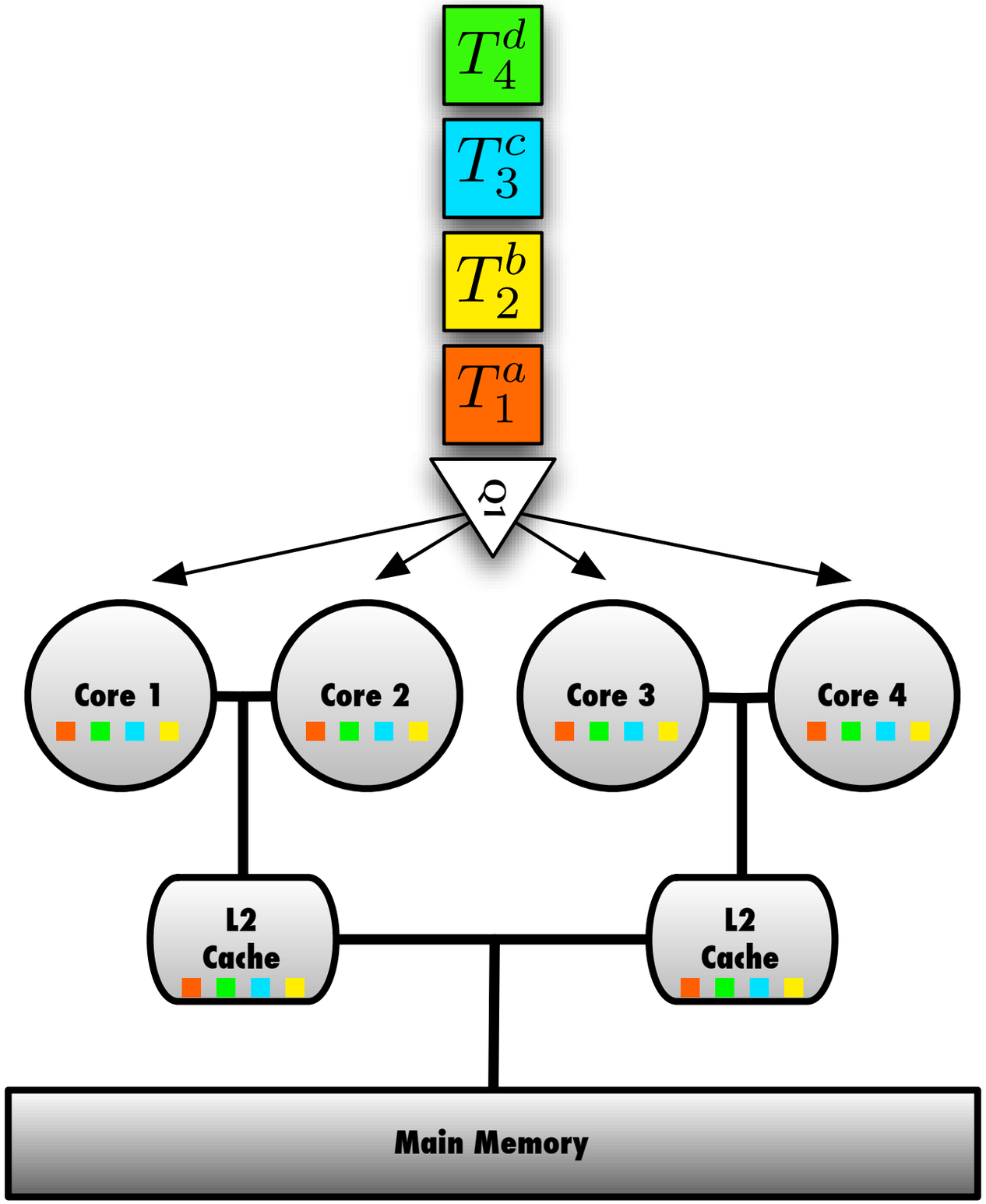
Bin packing

Even though there is sufficient total capacity, the last task cannot be placed.

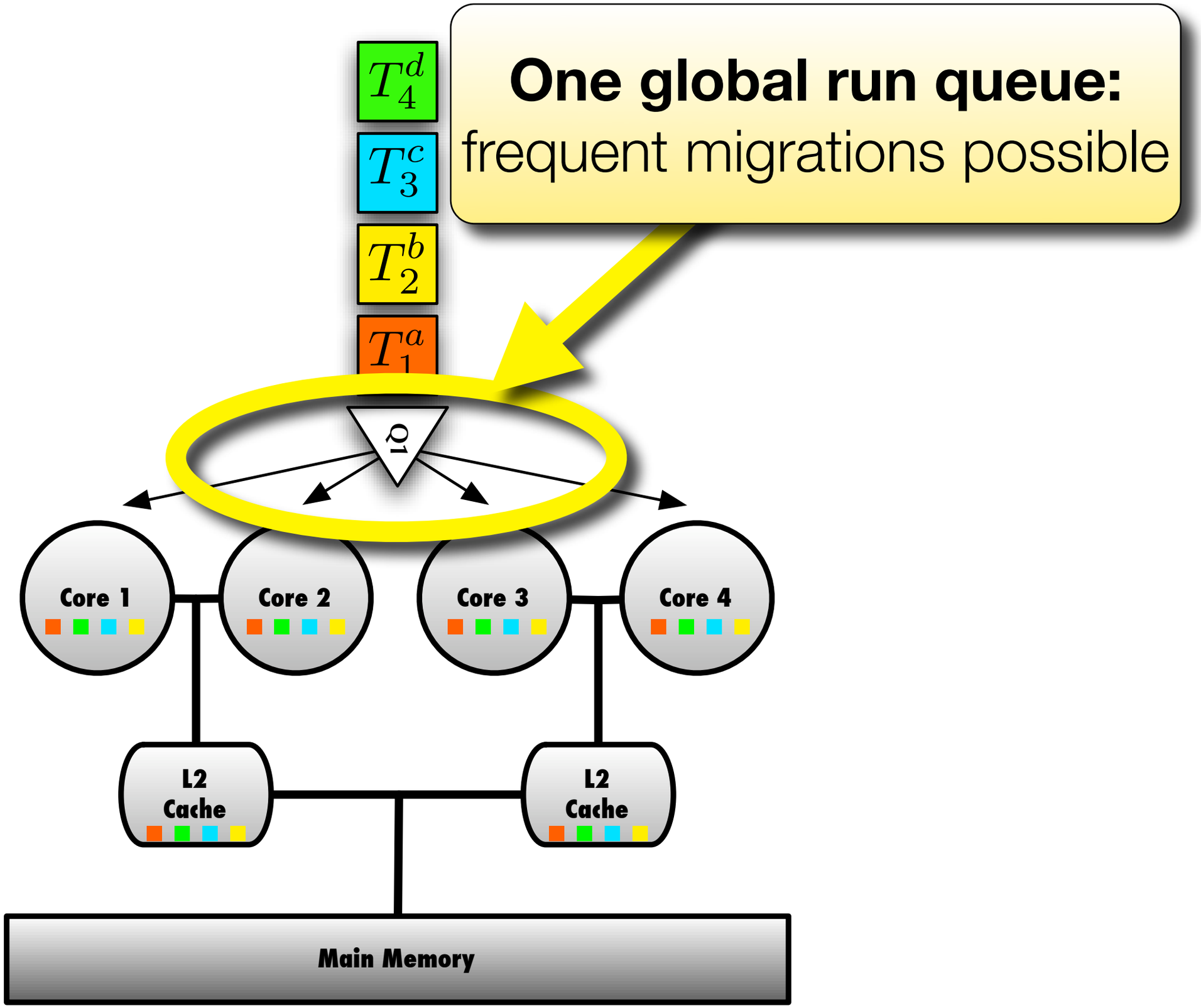
With partitioned scheduling, up to m/2 utilization may be wasted.



Global Scheduling



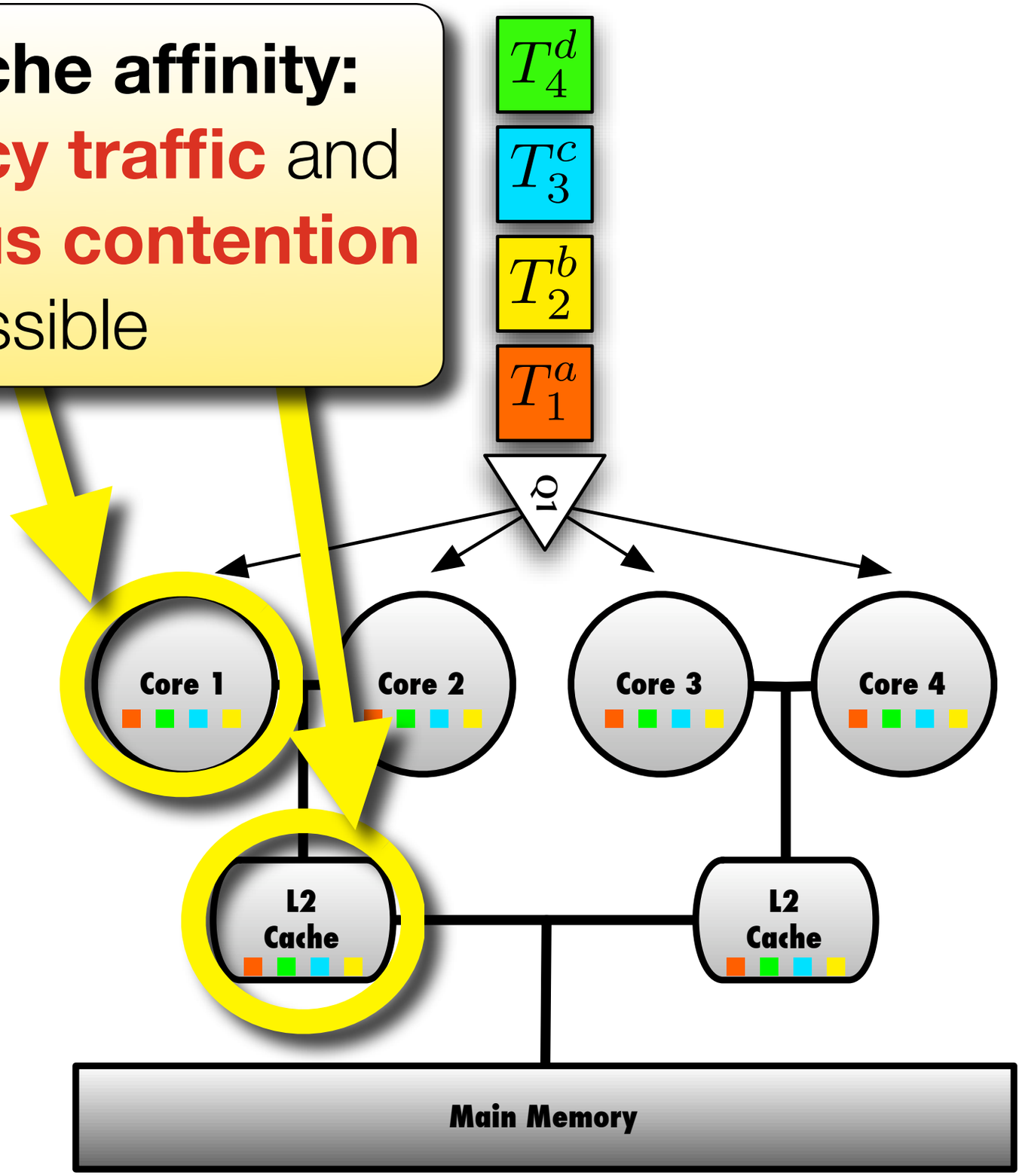
Global Scheduling



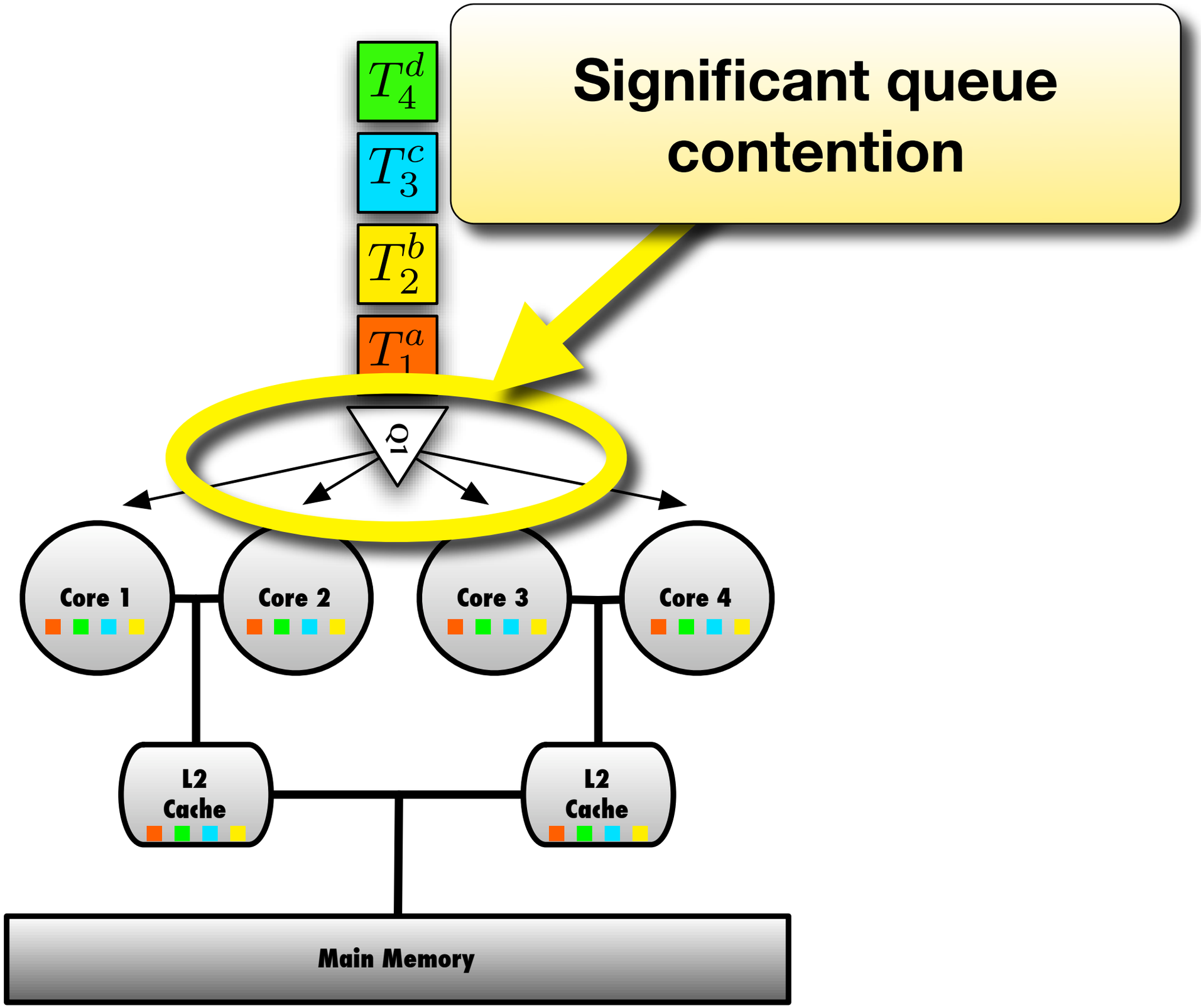
One global run queue:
frequent migrations possible

Global Scheduling

Weak cache affinity:
consistency traffic and
memory bus contention
possible

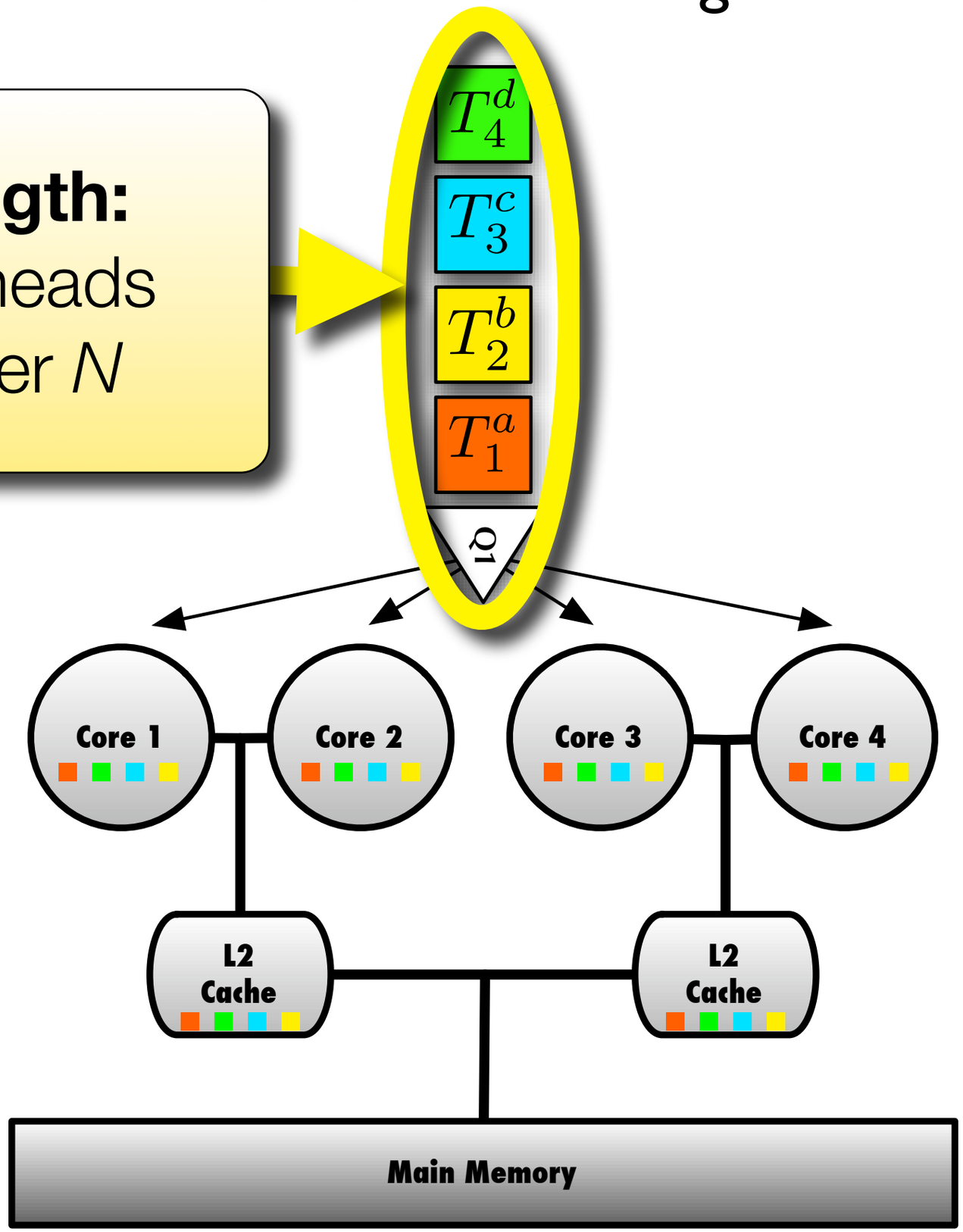


Global Scheduling



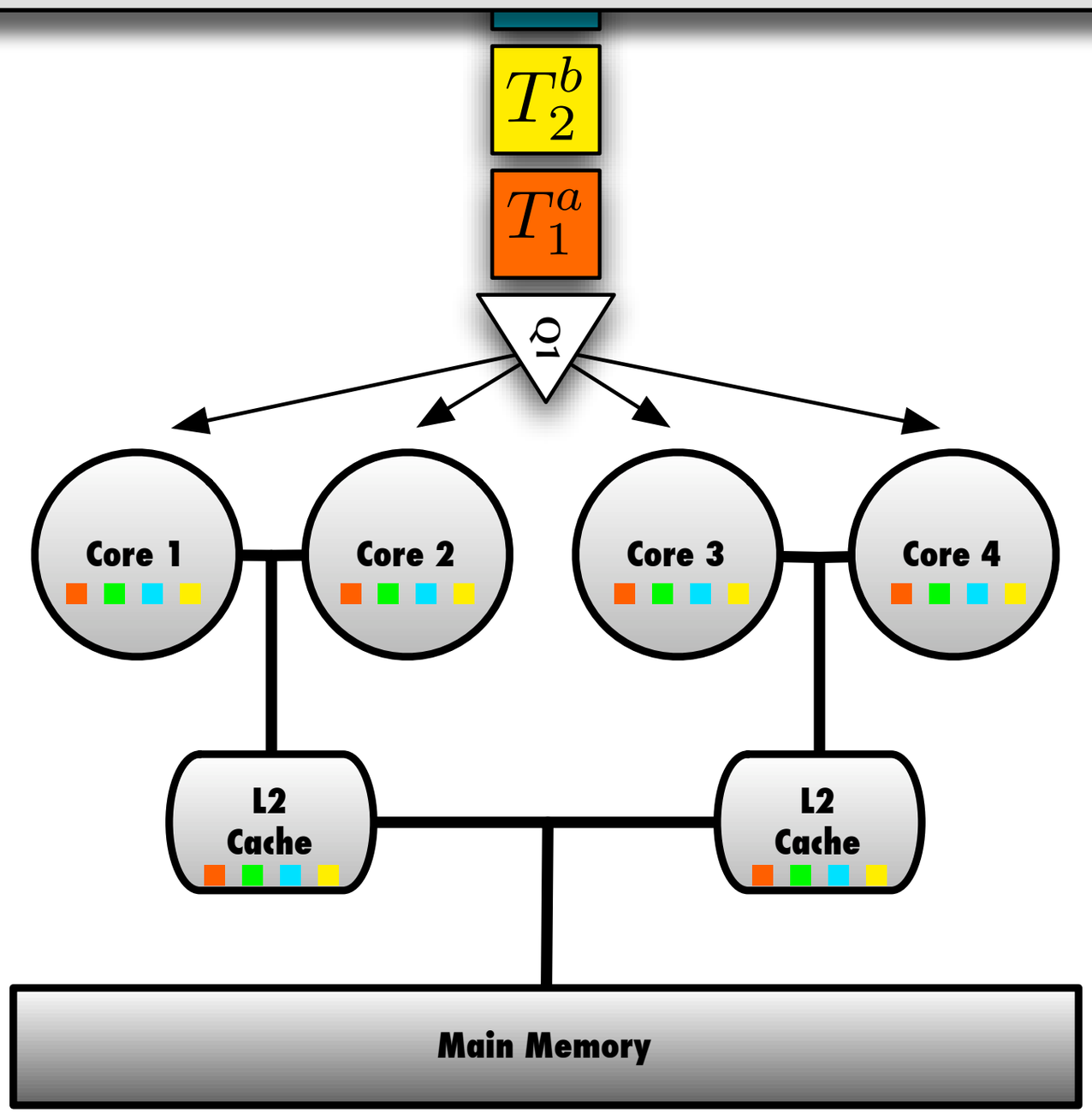
Global Scheduling

Queue length:
higher overheads
due to larger *N*



Global Scheduling

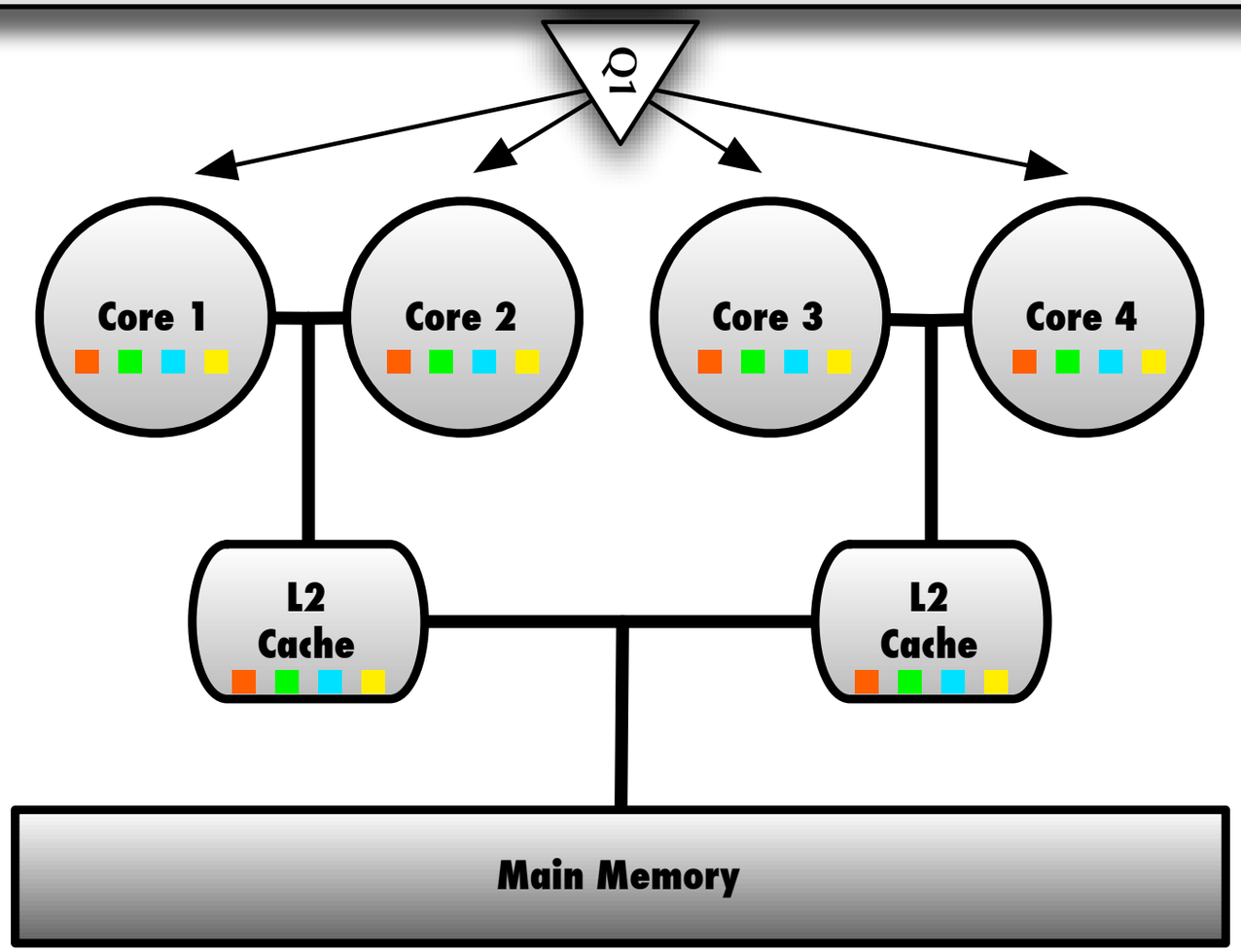
But: no bin-packing required!



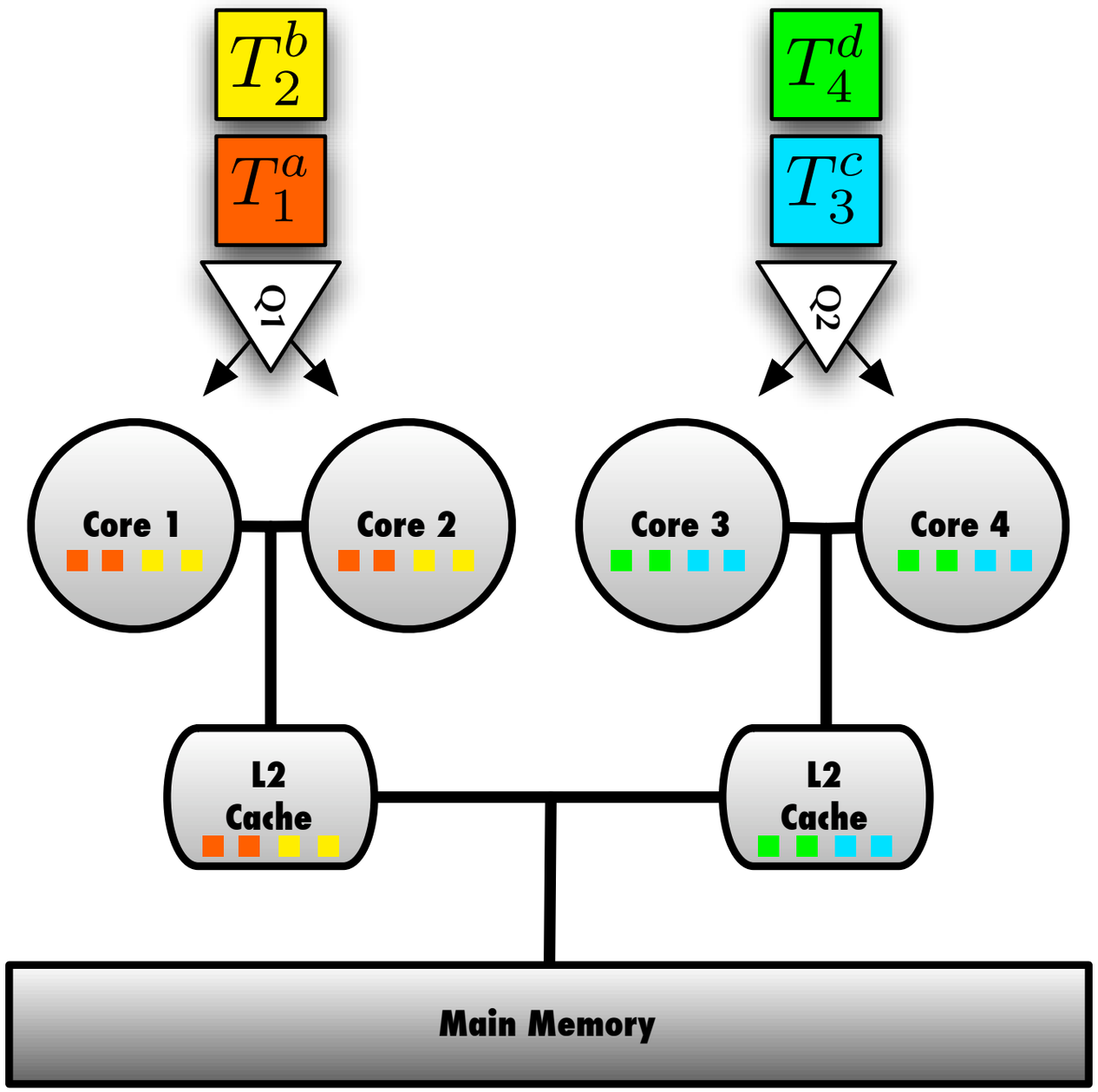
Global Scheduling

But: no bin-packing required!

All multiprocessor real-time scheduling algorithms that have been proven *optimal* are global.

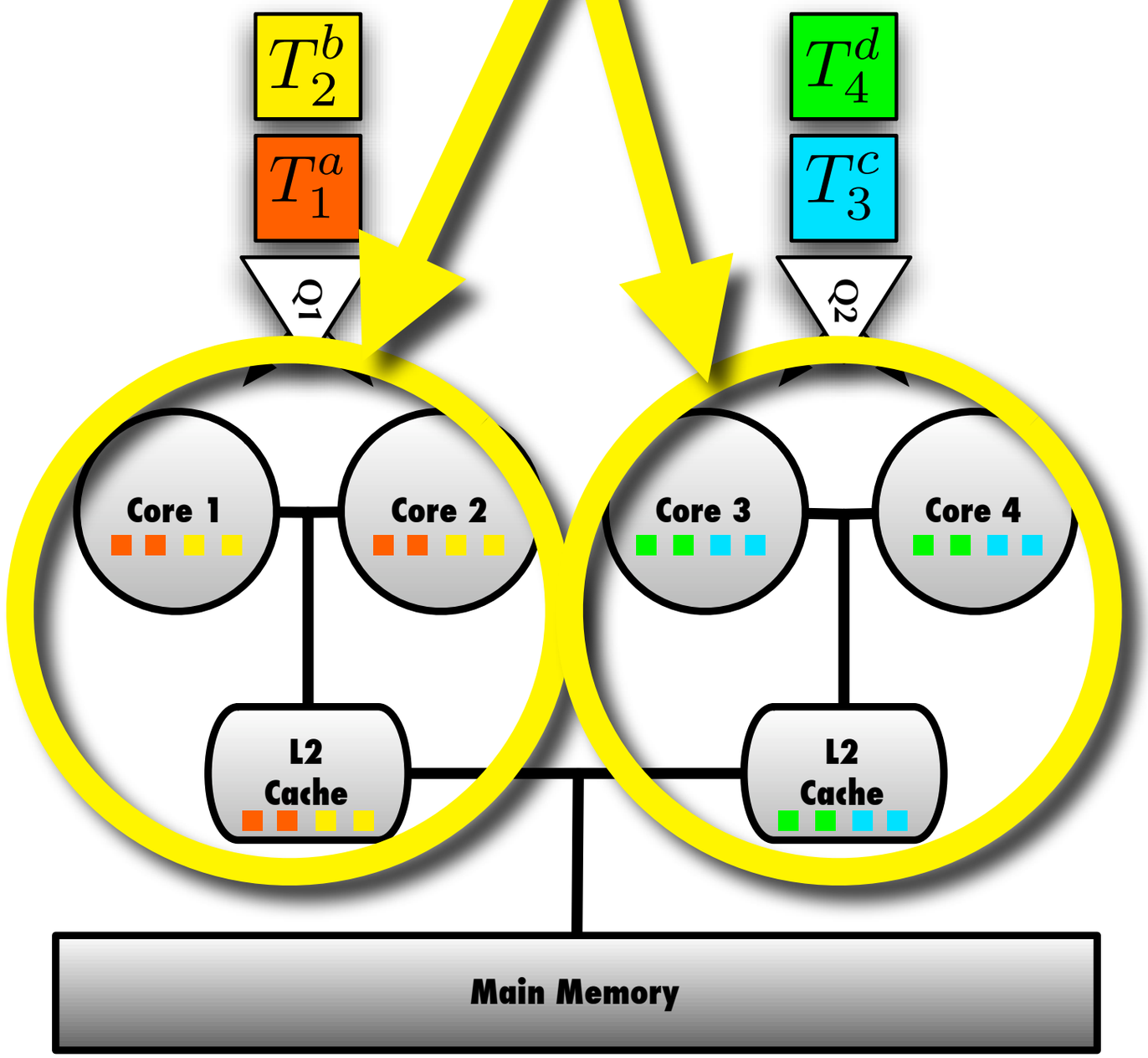


Clustered Scheduling



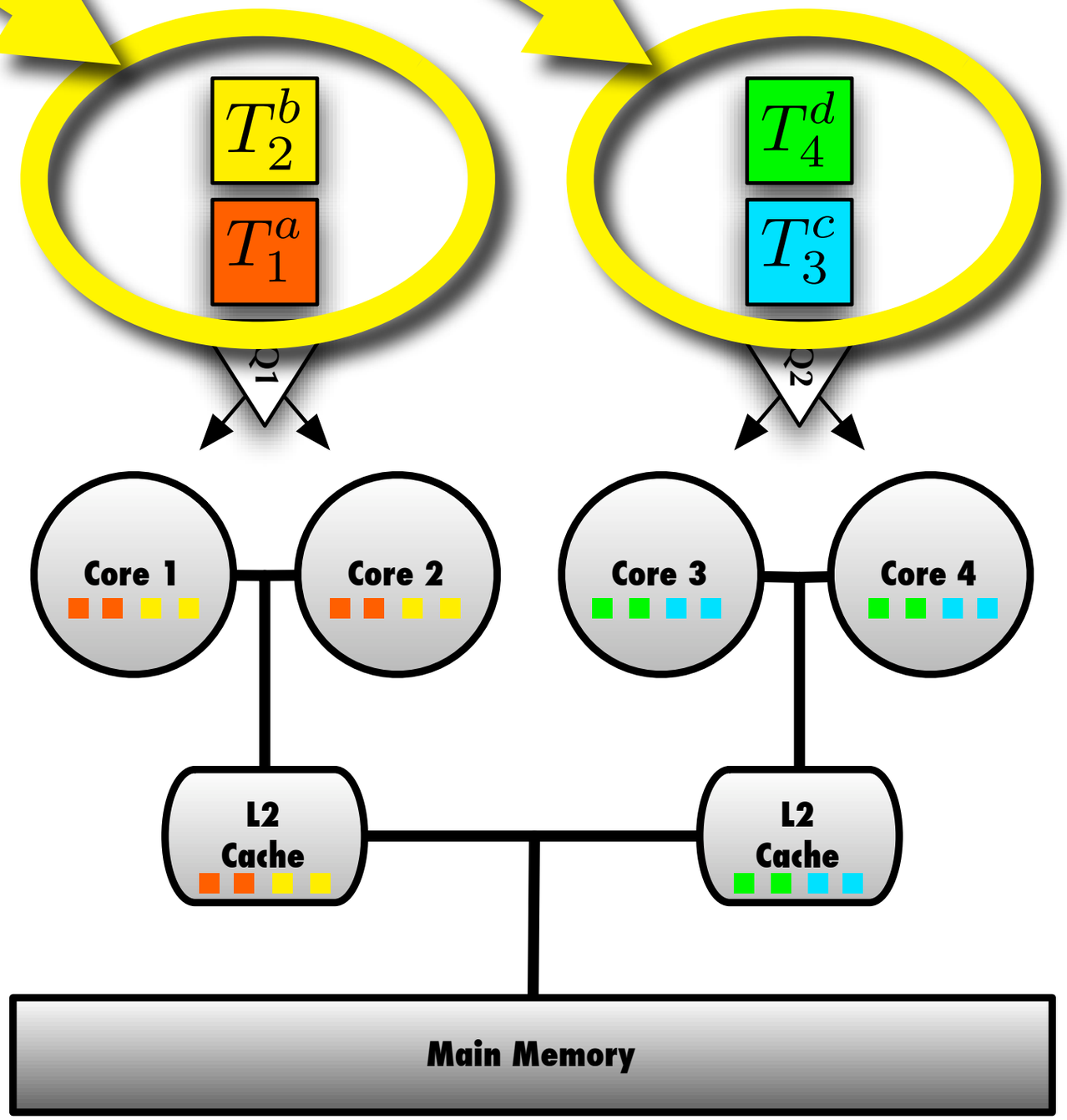
Clustered Scheduling

Group cores by shared caches

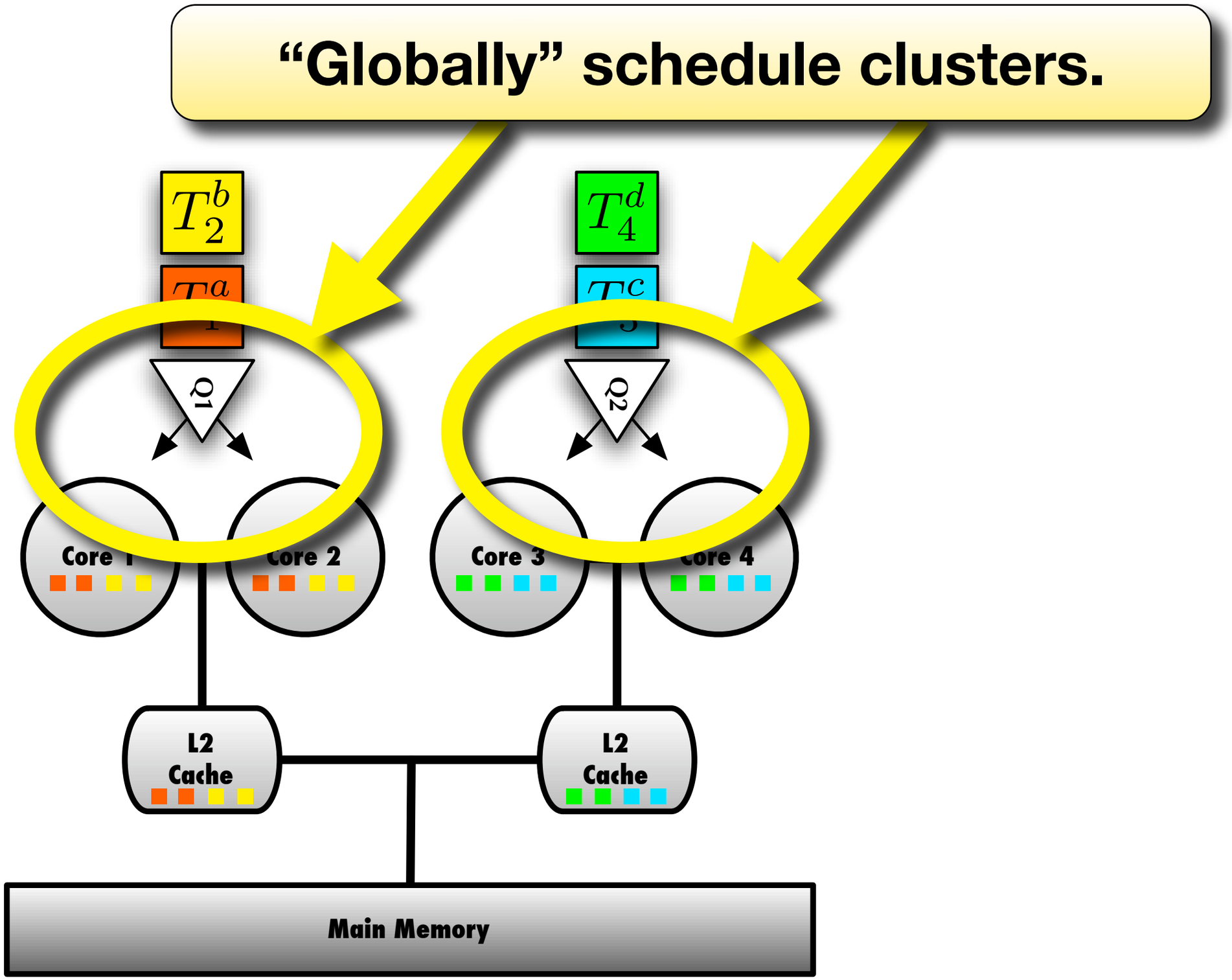


Clustered Scheduling

Statically assign tasks to clusters.

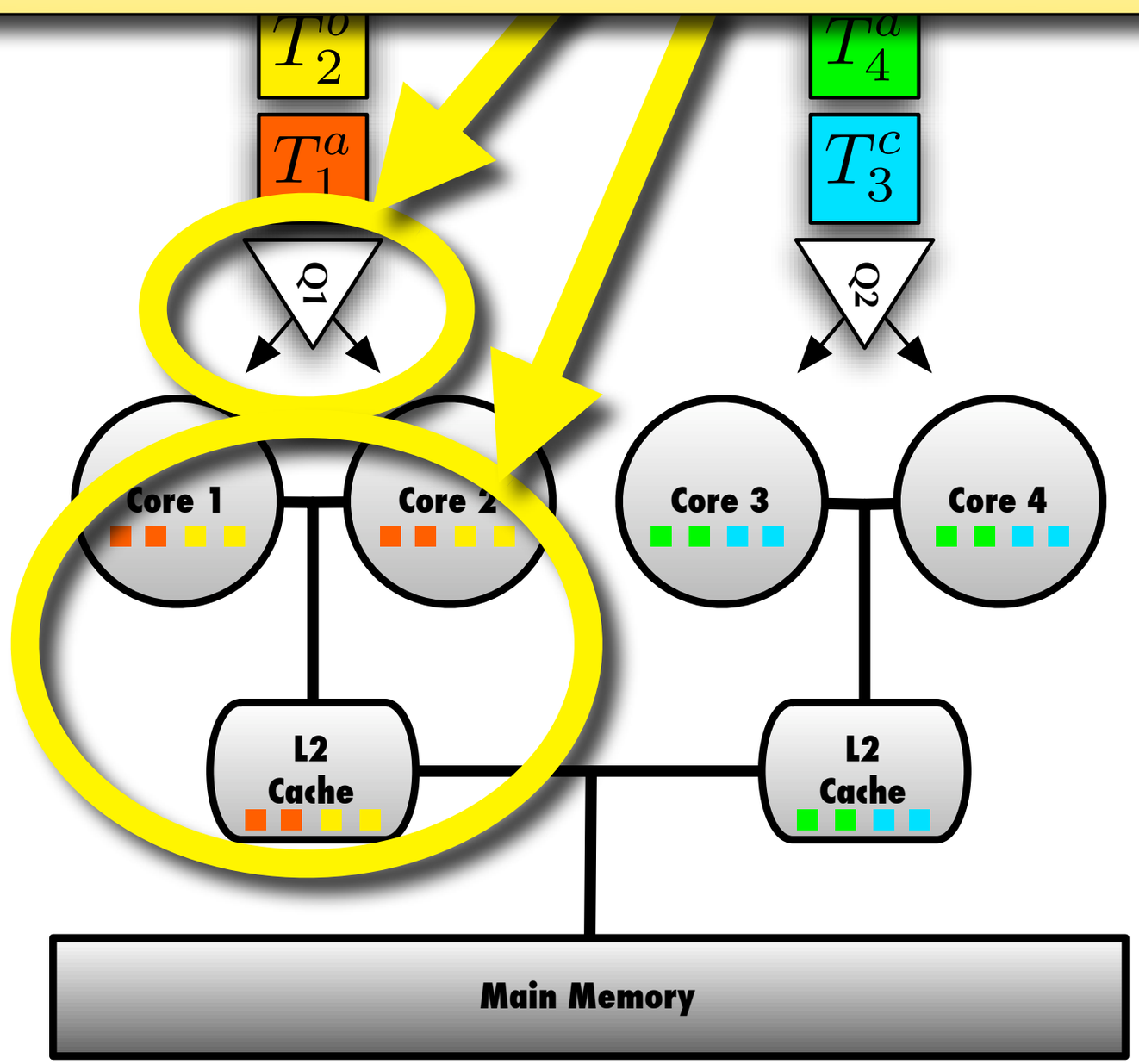


Clustered Scheduling

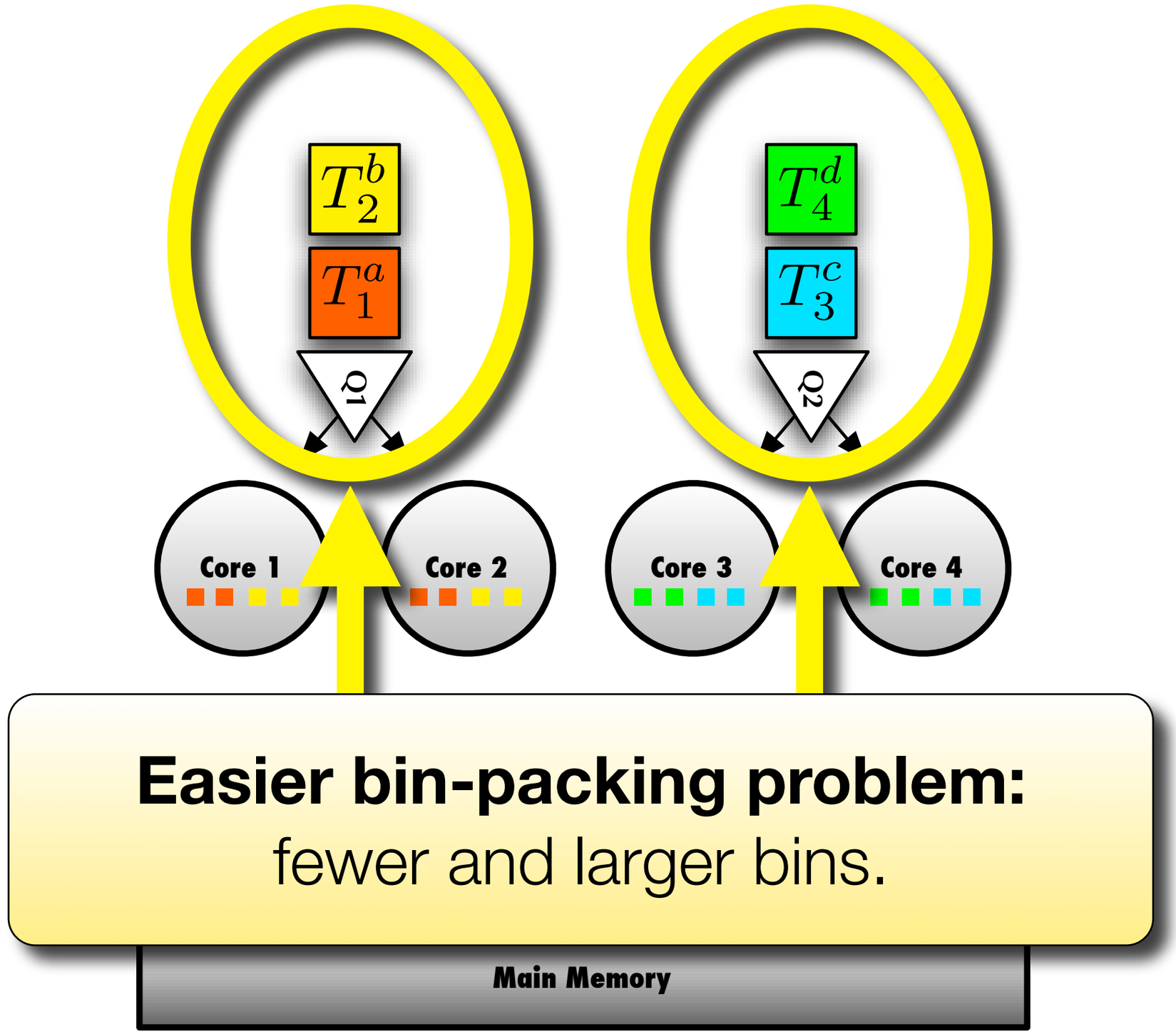


Clustered Scheduling

Less contention, better affinity
than pure global scheduling.



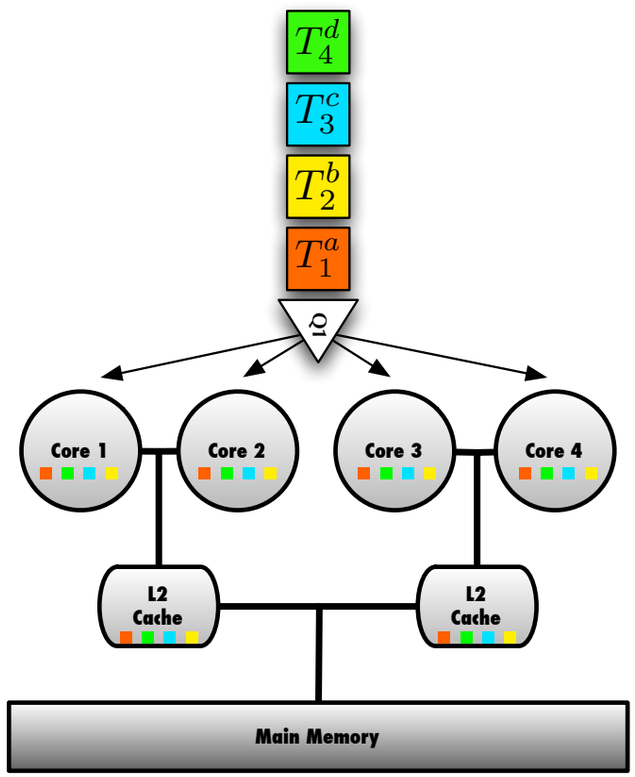
Clustered Scheduling

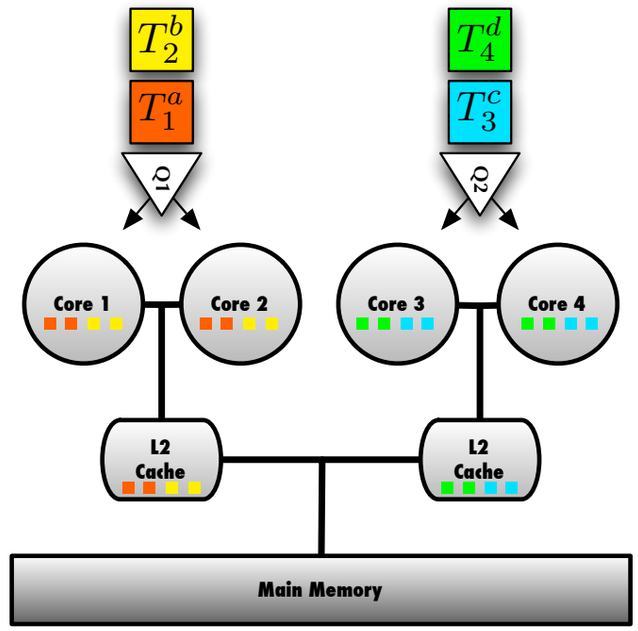


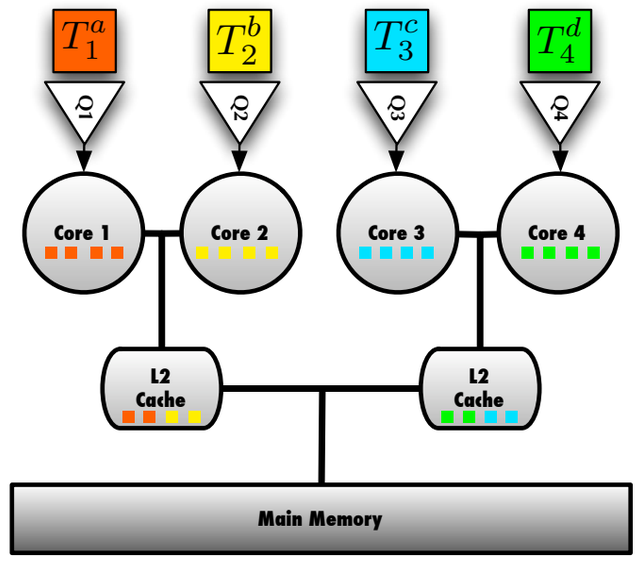
Easier bin-packing problem:
fewer and larger bins.

Relevant Scheduling Algorithms

global	clustered	partitioned

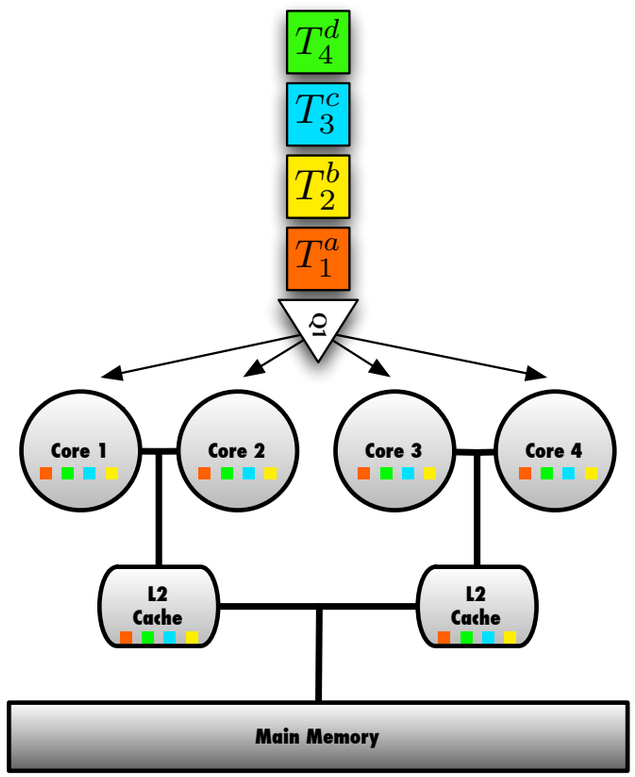


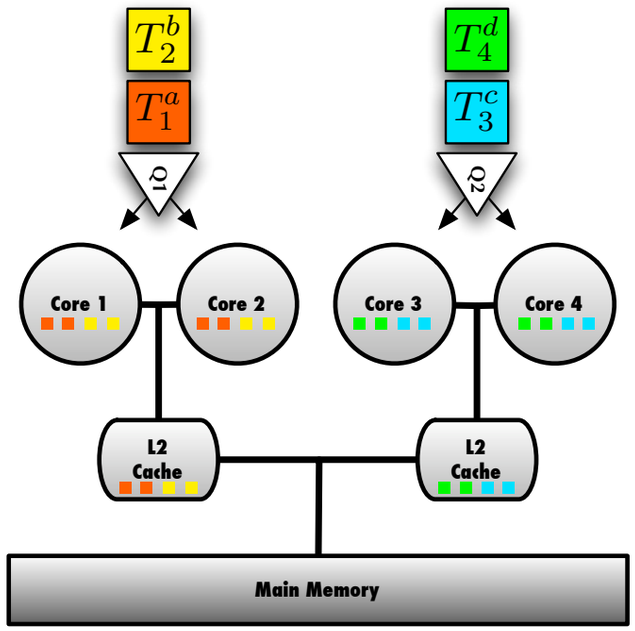


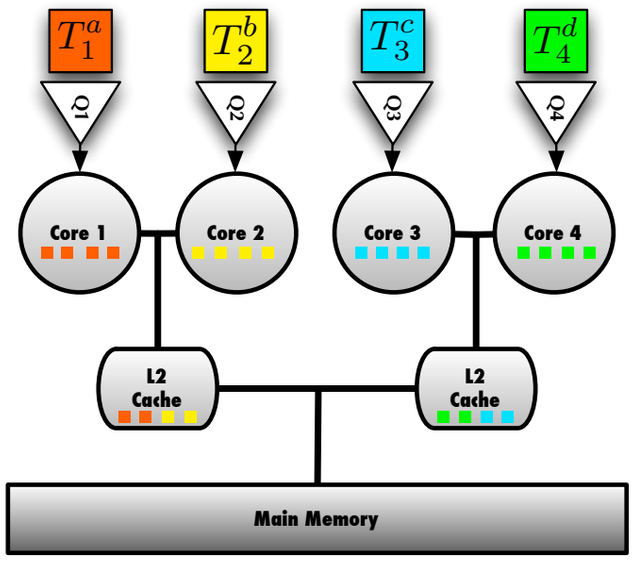


Relevant Scheduling Algorithms

global	clustered	partitioned
G-EDF	**C-EDF**	**P-EDF**

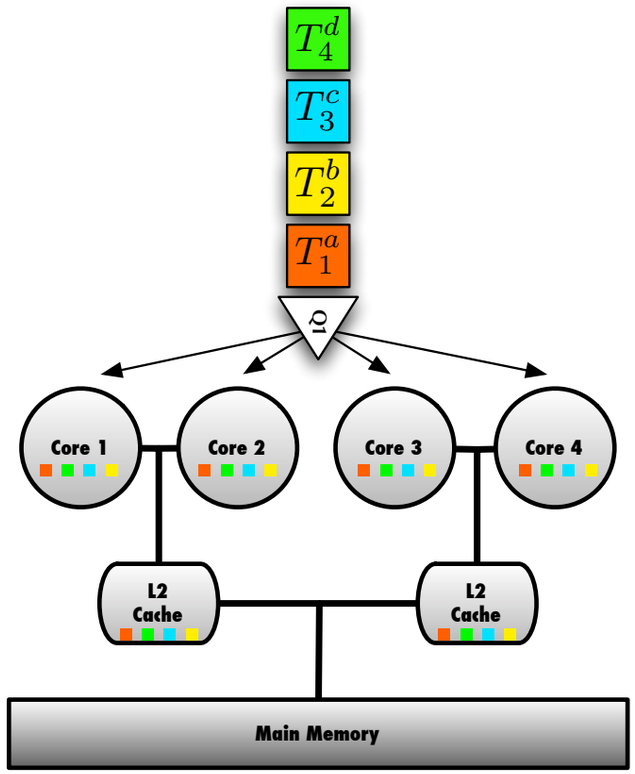


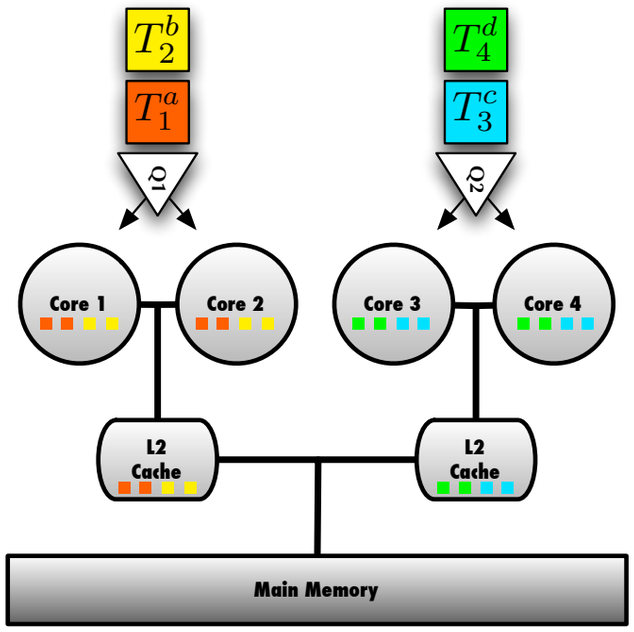


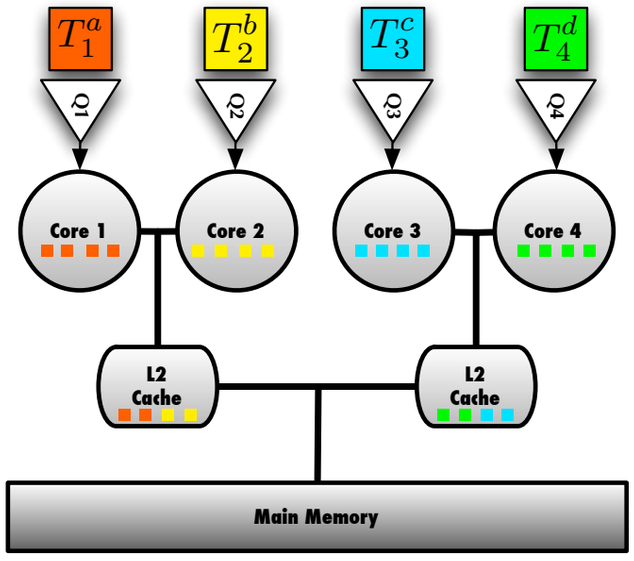


Relevant Scheduling Algorithms

global	clustered	partitioned
G-EDF	**C-EDF**	**P-EDF**
PFAIR	**C-PFAIR**	

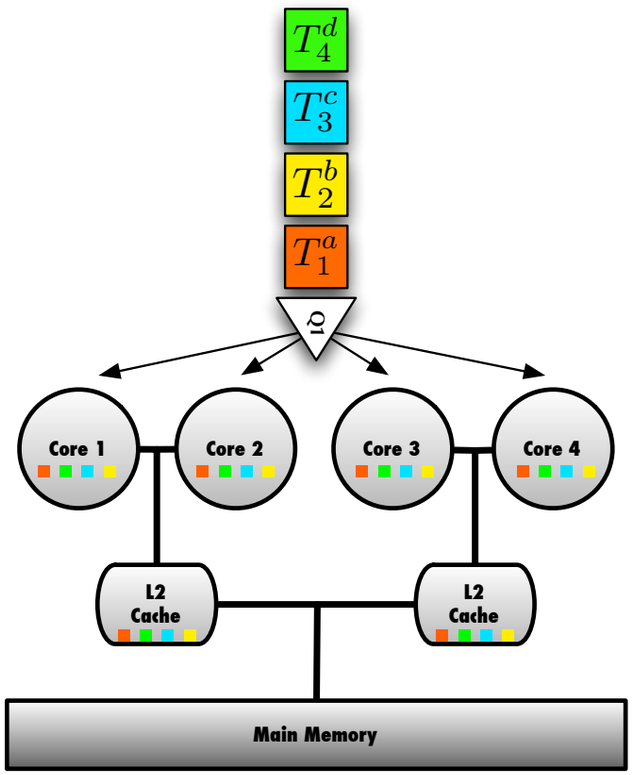


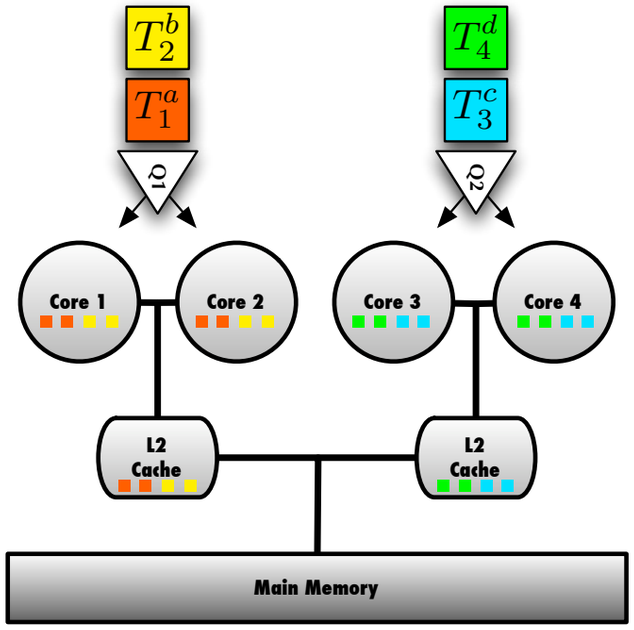


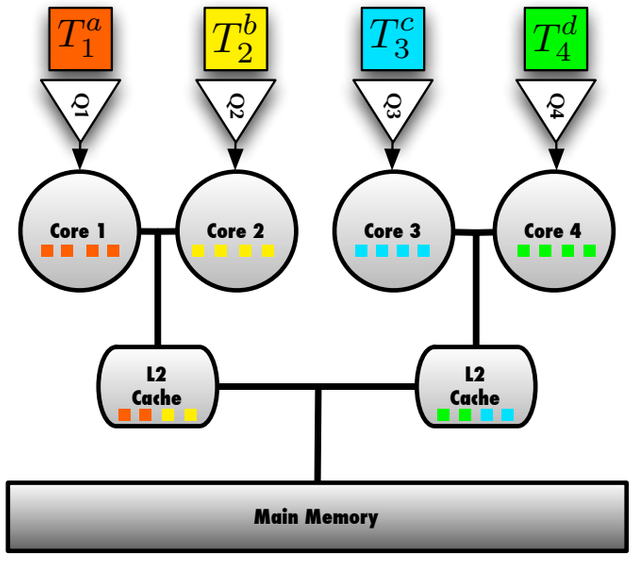


Relevant Scheduling Algorithms

global	clustered	partitioned
G-EDF	**C-EDF**	**P-EDF**
PFAIR	**C-PFAIR**	**P-SP**



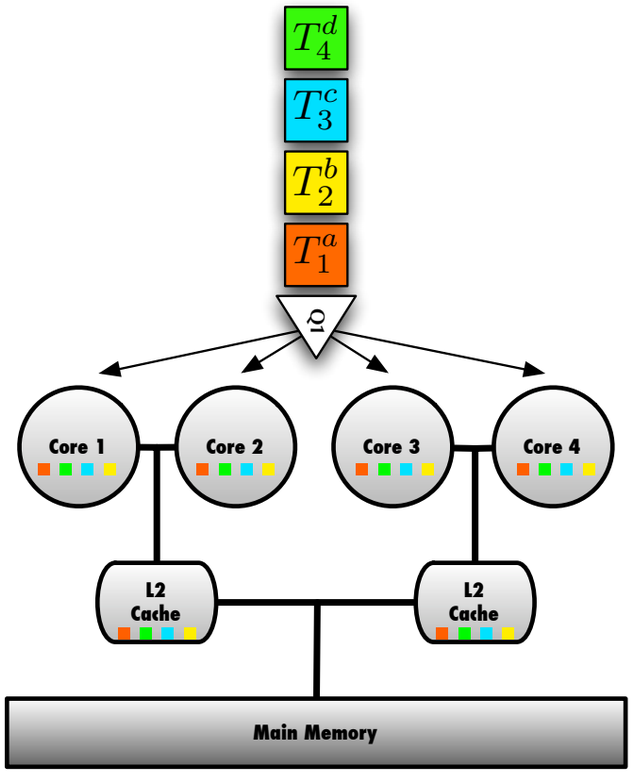


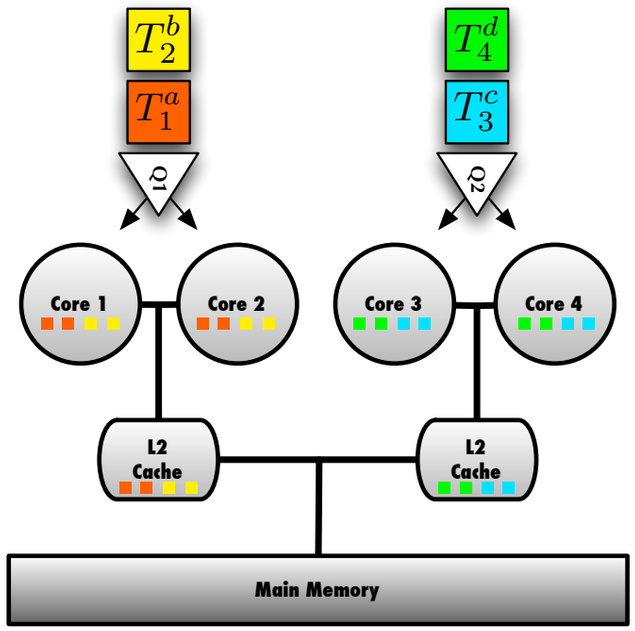


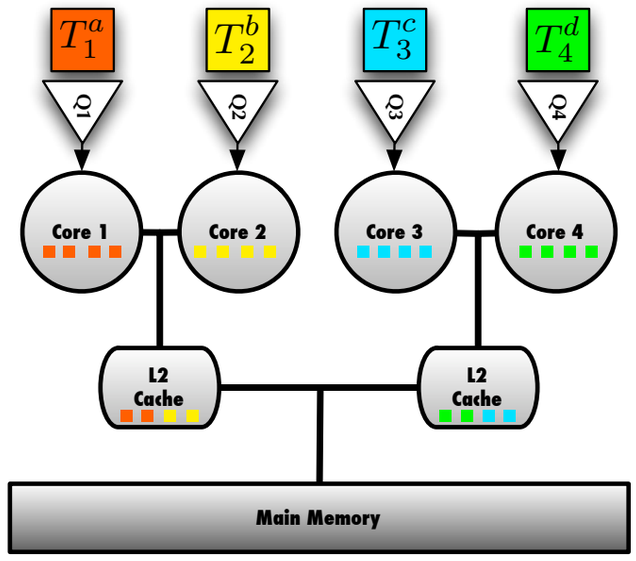
Theory: use global

only global algorithms do not require utilization caps

global	clustered	partitioned
G-EDF	**C-EDF**	**P-EDF**
PFAIR	**C-PFAIR**	**P-SP**







Theory: use global

only global algorithms do not require utilization caps

global | clustered | partitioned

G-EDF

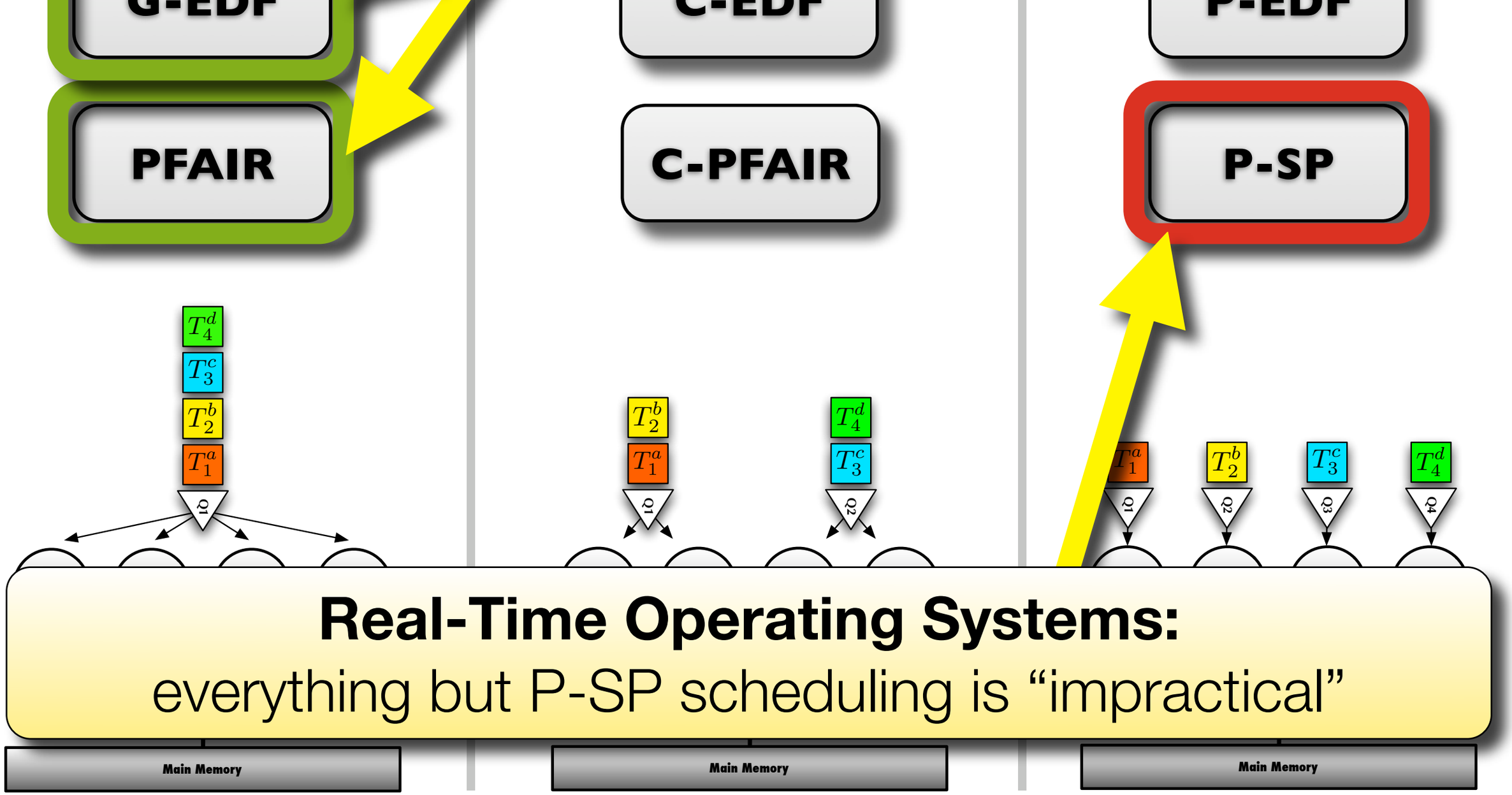
PFAIR

C-EDF

C-PFAIR

P-EDF

P-SP



Real-Time Operating Systems:

everything but P-SP scheduling is “impractical”