Scheduling and Analys of Limited-Preemptive Modable Gang Tasks

Joan Marcè i Igual Geoffrey Nelissen Mitra Nasri Paris Panagiotou

24th of February, 2020

What is gang?

- Parallel threads executed together as a “gang”
- Execution does not start until there are enough free cores

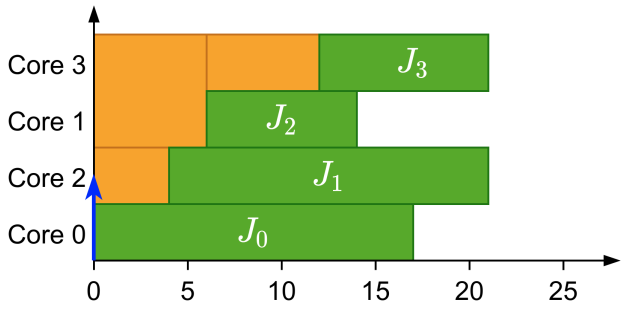
First of all, let’s explain what is gang scheduling.

It’s the execution of multiple parallel threads together as a “gang”. In these threads the execution does not start until there are enough cores to execute them together.
Let’s see an example:

1. We have these jobs assigned to different cores.
2. If we release them as a gang job then we should pack them like a single task. Obtaining the following group. <click>
3. Then,<click> we have obtained the gang task result of merging the jobs

What is gang?

- Parallel threads executed together as a “gang”
- Execution does not start until there are enough free cores



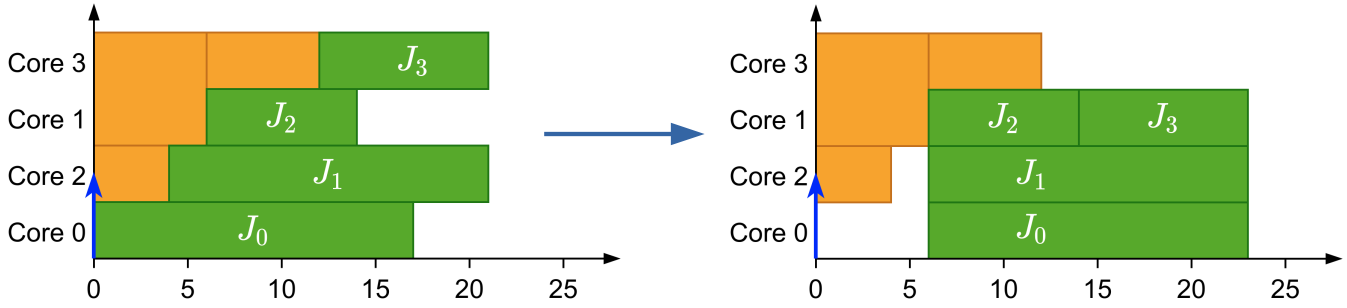
First of all, let’s explain what is gang scheduling.

It’s the execution of multiple parallel threads together as a “gang”. In these threads the execution does not start until there are enough cores to execute them together.
Let’s see an example:

1. We have these jobs assigned to different cores.
2. If we release them as a gang job then we should pack them like a single task. Obtaining the following group. <click>
3. Then,<click> we have obtained the gang task result of merging the jobs

What is gang?

- Parallel threads executed together as a “gang”
- Execution does not start until there are enough free cores



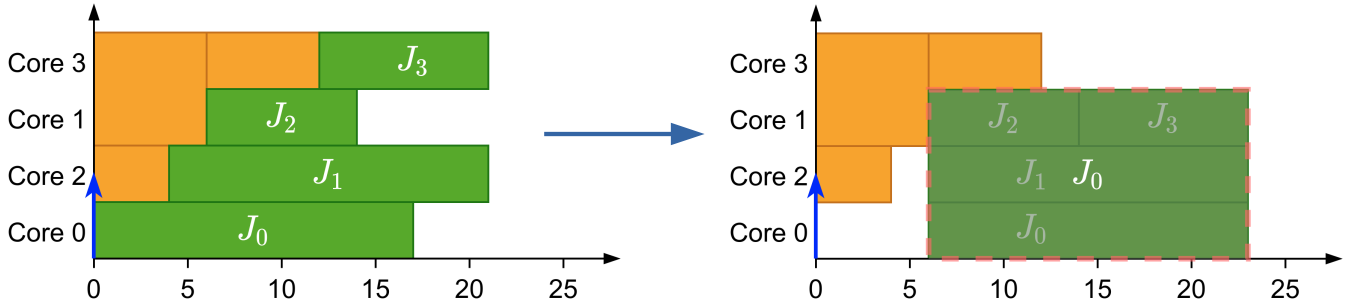
First of all, let’s explain what is gang scheduling.

It’s the execution of multiple parallel threads together as a “gang”. In these threads the execution does not start until there are enough cores to execute them together.
Let’s see an example:

1. We have these jobs assigned to different cores.
2. If we release them as a gang job then we should pack them like a single task. Obtaining the following group. <click>
3. Then,<click> we have obtained the gang task result of merging the jobs

What is gang?

- Parallel threads executed together as a “gang”
- Execution does not start until there are enough free cores



First of all, let’s explain what is gang scheduling.

It’s the execution of multiple parallel threads together as a “gang”. In these threads the execution does not start until there are enough cores to execute them together.
Let’s see an example:

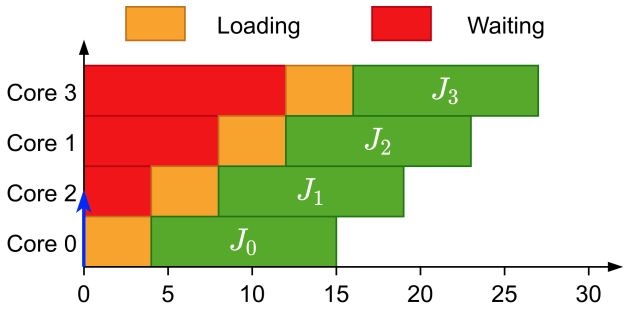
1. We have these jobs assigned to different cores.
2. If we release them as a gang job then we should pack them like a single task. Obtaining the following group. <click>
3. Then,<click> we have obtained the gang task result of merging the jobs

Why gang?

- Allows synchronization
- Avoids overhead when loading initial data

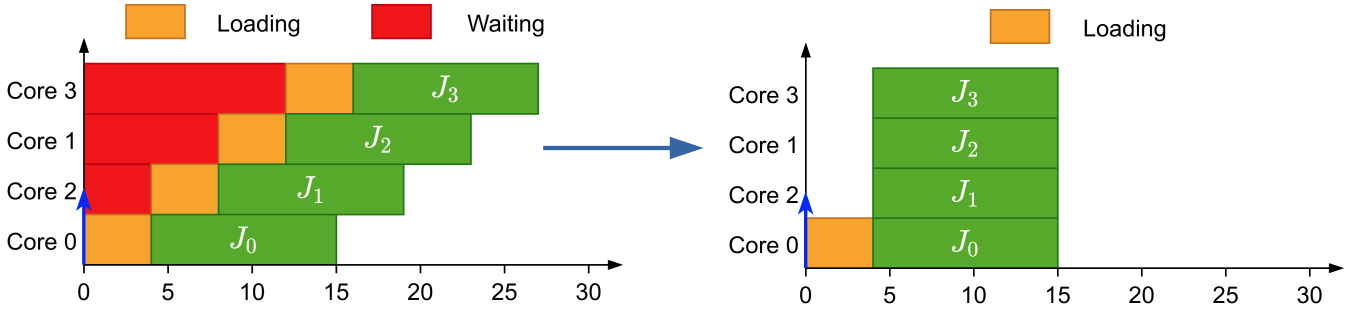
Why gang?

- Allows synchronization
- Avoids overhead when loading initial data



Why gang?

- Allows synchronization
- Avoids overhead when loading initial data

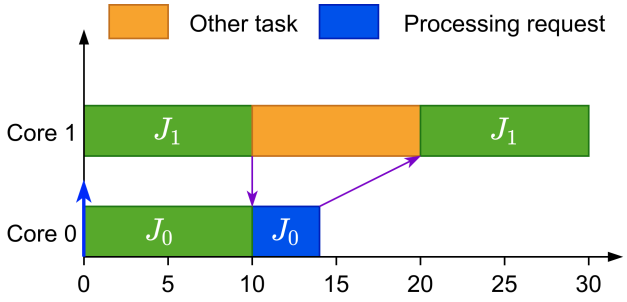


Why gang?

- Allows synchronization
- Avoids overhead when loading initial data

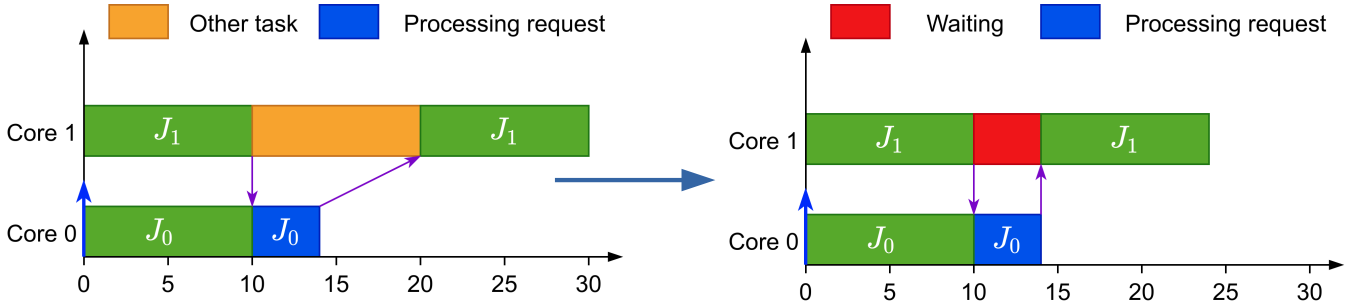
Why gang?

- Allows synchronization
- Avoids overhead when loading initial data



Why gang?

- Allows synchronization
- Avoids overhead when loading initial data



Types of gang

- **Rigid**: number of cores set by programmer
- **Moldable**: number of cores assigned during scheduling
- **Malleable**: number of cores can change during runtime

We can have three types of gang scheduling:

- Rigid gang, where the number of cores is set by the programmer and fixed during execution
- Moldable gang, where the number of cores can be in a range of values and the actual chosen value is decided when scheduling the task
- Malleable gang, where the number of cores can vary during the execution of the job

Previous work

Rigid gang solutions

Moldable gang solutions

Malleable gang solutions

Schedulability analysis

Our work

LPMRGS