Scheduling pipelined applications: models, algorithms and complexity

Anne Benoit
GRAAL team, LIP, École Normale Supérieure de Lyon, France

ASTEC meeting in Les Plantiers, France
June 2, 2009

Introduction and motivation

- Schedule an application onto a computational platform, with some criteria to optimize
- Target application
  - Streaming application (workflow, pipeline): several data sets are processed by a set of tasks (or pipeline stages)
  - Linear chain application: linear dependencies between tasks
  - Extensions: filtering services, general DAGs, more complex applications, ...
- Target platform
  - ranking from fully homogeneous to fully heterogeneous
  - completely interconnected, subject to failures
  - emphasis on different communication models (overlap or not, one- vs multi-port)
- Optimization criteria
  - period (inverse of throughput) and latency (execution time)
  - reliability, and also energy, stretch, ...

Introduction and motivation

- Schedule an application onto a computational platform, with some criteria to optimize
- Target application
  - Streaming application (workflow, pipeline): several data sets are processed by a set of tasks (or pipeline stages)
  - Linear chain application: linear dependencies between tasks
  - Extensions: filtering services, general DAGs, more complex applications, ...
- Target platform
  - ranking from fully homogeneous to fully heterogeneous
  - completely interconnected, subject to failures
  - emphasis on different communication models (overlap or not, one- vs multi-port)
- Optimization criteria
  - period (inverse of throughput) and latency (execution time)
  - reliability, and also energy, stretch, ...

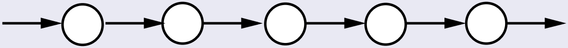
Introduction and motivation

- Schedule an application onto a computational platform, with some criteria to optimize
- Target application
  - Streaming application (workflow, pipeline): several data sets are processed by a set of tasks (or pipeline stages)
  - Linear chain application: linear dependencies between tasks
  - Extensions: filtering services, general DAGs, more complex applications, ...
- Target platform
  - ranking from fully homogeneous to fully heterogeneous
  - completely interconnected, subject to failures
  - emphasis on different communication models (overlap or not, one- vs multi-port)
- Optimization criteria
  - period (inverse of throughput) and latency (execution time)
  - reliability, and also energy, stretch, ...

Introduction and motivation

- Schedule an application onto a computational platform, with some criteria to optimize
- Target application
  - Streaming application (workflow, pipeline): several data sets are processed by a set of tasks (or pipeline stages)
  - Linear chain application: linear dependencies between tasks
  - Extensions: filtering services, general DAGs, more complex applications, ...
- Target platform
  - ranking from fully homogeneous to fully heterogeneous
  - completely interconnected, subject to failures
  - emphasis on different communication models (overlap or not, one- vs multi-port)
- Optimization criteria
  - period (inverse of throughput) and latency (execution time)
  - reliability, and also energy, stretch, ...

Linear chain pipelined applications



Several consecutive data sets enter the application graph.

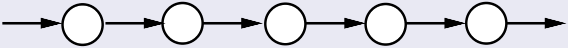
Multi-criteria to optimize?

Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

Latency $\mathcal{L}$: maximal time elapsed between beginning and end of execution of a data set

Reliability: inverse of $\mathcal{F}$, probability of failure of the application (i.e. some data sets will not be processed)

Linear chain pipelined applications



Several consecutive data sets enter the application graph.

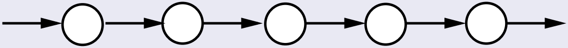
Multi-criteria to optimize?

Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

Latency $\mathcal{L}$: maximal time elapsed between beginning and end of execution of a data set

Reliability: inverse of $\mathcal{F}$, probability of failure of the application (i.e. some data sets will not be processed)

Linear chain pipelined applications



Several consecutive data sets enter the application graph.

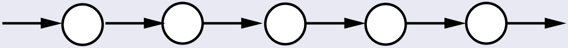
Multi-criteria to optimize?

Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

Latency $\mathcal{L}$: maximal time elapsed between beginning and end of execution of a data set

Reliability: inverse of $\mathcal{F}$, probability of failure of the application (i.e. some data sets will not be processed)

Linear chain pipelined applications



Several consecutive data sets enter the application graph.

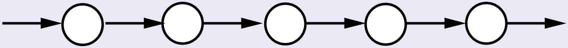
Multi-criteria to optimize?

Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

Latency $\mathcal{L}$: maximal time elapsed between beginning and end of execution of a data set

Reliability: inverse of $\mathcal{F}$, probability of failure of the application (i.e. some data sets will not be processed)

Linear chain pipelined applications



Several consecutive data sets enter the application graph.

Multi-criteria to optimize?

Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

Latency $\mathcal{L}$: maximal time elapsed between beginning and end of execution of a data set

Reliability: inverse of $\mathcal{F}$, probability of failure of the application (i.e. some data sets will not be processed)

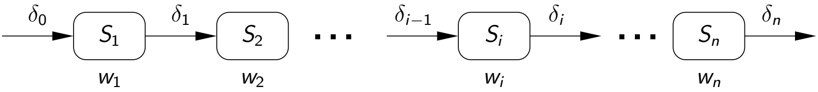
Outline

1. Models
   - Application model
   - Platform and communication models
   - Multi-criteria mapping problems
2. Complexity results
   - Mono-criterion problems
   - Bi-criteria problems
3. Conclusion

Outline

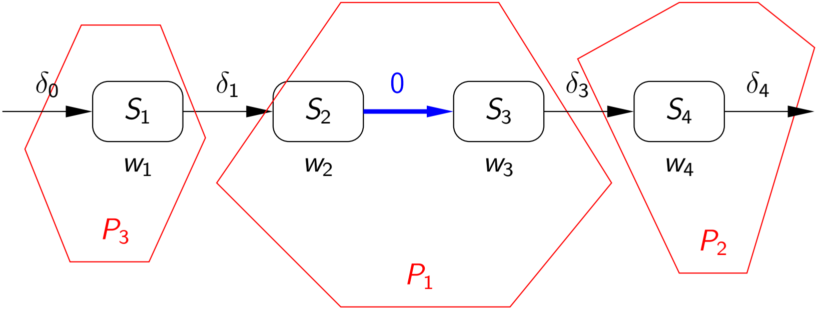
Application model

- Set of n application stages
- Computation cost of stage S_i: w_i
- Pipelined: each data set must be processed by all stages
- Linear dependencies between stages

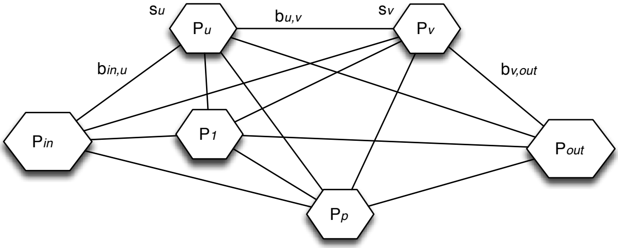


Application model: communication costs

- Two dependent stages $S_i \rightarrow S_{i+1}$:
  data must be transferred from S_i to S_{i+1}
- Fixed data size δ_i, communication cost to pay only if S_i and S_{i+1} are mapped onto different processors
  (i.e., no cost on blue arrow in the example)



Platform model



- p + 2 processors P_u, $0 \leq u \leq \mathrm{p}+1$
- $P_0 = P_{in}$: input data – $P_{p+1} = P_{out}$: output data
- P_1 to P_p: fully interconnected (clique)
- s_u: speed of processor P_u, $1 \leq u \leq p$, liner cost model
- bidirectional link $\mathrm{link}_{u,v} : P_u \rightarrow P_v$, bandwidth $b_{u,v}$
- B_u^i / B_u^o: input/output network card capacity

Platform model: classification

Fully Homogeneous – Identical processors ($s_u = s$) and homogeneous communication devices ($b_{u,v} = b, \mathrm{B}^i_u = \mathrm{B}^i, \mathrm{B}^o_u = \mathrm{B}^o$): typical parallel machines

Communication Homogeneous – Homogeneous communication devices but different-speed processors ($s_u \neq s_v$): networks of workstations, clusters

Fully Heterogeneous – Fully heterogeneous architectures: hierarchical platforms, grids

Platform model: unreliable processors

- f_u: failure probability of processor P_u
  - independent of the duration of the application: global indicator of processor reliability
  - steady-state execution: loan/rent resources, cycle-stealing
  - fail-silent/fail-stop, no link failures (use different paths)
- *Failure Homogeneous*– Identically reliable processors ($f_u = f_v$), natural with *Fully Homogeneous*
- *Failure Heterogeneous* – Different failure probabilities ($f_u \neq f_v$), natural with *Communication Homogeneous* and *Fully Heterogeneous*

Platform model: unreliable processors

- f_u: failure probability of processor P_u
  - independent of the duration of the application: global indicator of processor reliability
  - steady-state execution: loan/rent resources, cycle-stealing
  - fail-silent/fail-stop, no link failures (use different paths)

- *Failure Homogeneous*– Identically reliable processors ($f_u = f_v$), natural with *Fully Homogeneous*
- *Failure Heterogeneous* – Different failure probabilities ($f_u \neq f_v$), natural with *Communication Homogeneous* and *Fully Heterogeneous*

Platform model: communications, a bit of history

Classical communication model in scheduling works:
macro-dataflow model

$$cost(T, T') = \begin{cases} 0 & \text{if } alloc(T) = alloc(T') \\ comm(T, T') & \text{otherwise} \end{cases}$$

- Task T communicates data to successor task T'
- $alloc(T)$: processor that executes T; $comm(T, T')$: defined by the application specification
- Two main assumptions:
  - (i) communication can occur as soon as data are available
  - (ii) no contention for network links
- (i) is reasonable, (ii) assumes infinite network resources!

Platform model: communications, a bit of history

Classical communication model in scheduling works:
macro-dataflow model

$$cost(T, T') = \begin{cases} 0 & \text{if } alloc(T) = alloc(T') \\ comm(T, T') & \text{otherwise} \end{cases}$$

- Task T communicates data to successor task T'
- $alloc(T)$: processor that executes T; $comm(T, T')$: defined by the application specification
- Two main assumptions:
  - (i) communication can occur as soon as data are available
  - (ii) no contention for network links
- (i) is reasonable, (ii) assumes infinite network resources!

Platform model: communications, a bit of history

Classical communication model in scheduling works:
macro-dataflow model

$$cost(T, T') = \begin{cases} 0 & \text{if } alloc(T) = alloc(T') \\ comm(T, T') & \text{otherwise} \end{cases}$$

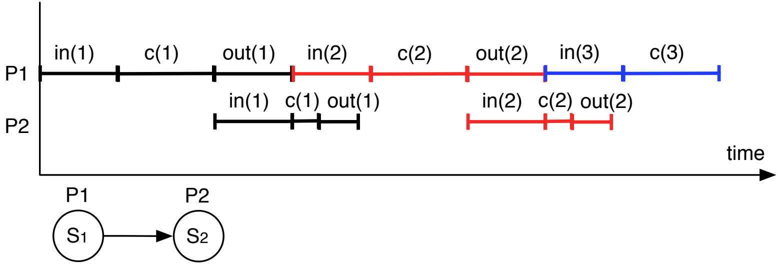
- Task T communicates data to successor task T'
- $alloc(T)$: processor that executes T; $comm(T, T')$: defined by the application specification
- Two main assumptions:
  - (i) communication can occur as soon as data are available
  - (ii) no contention for network links
- (i) is reasonable, (ii) assumes infinite network resources!

Platform model: one-port without overlap

- no overlap: at each time step, either computation or communication
- one-port: each processor can either send or receive to/from a single other processor any time step it is communicating

Platform model: one-port without overlap

- no overlap: at each time step, either computation or communication
- one-port: each processor can either send or receive to/from a single other processor any time step it is communicating

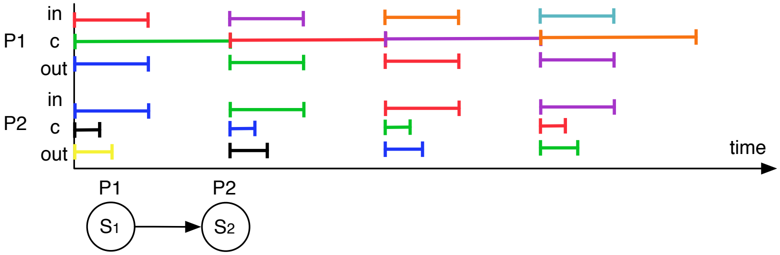


Platform model: bounded multi-port with overlap

- overlap: a processor can simultaneously compute and communicate
- bounded multi-port: simultaneous send and receive, but bound on the total outgoing/incoming communication (limitation of network card)

Platform model: bounded multi-port with overlap

- overlap: a processor can simultaneously compute and communicate
- bounded multi-port: simultaneous send and receive, but bound on the total outgoing/incoming communication (limitation of network card)



Platform model: communication models

- Multi-port: if several non-consecutive stages mapped onto a same processor, several concurrent communications
- Matches multi-threaded systems
- Fits well together with overlap

- One-port: radical option, where everything is serialized
- Natural to consider it without overlap

- Other communication models: more complicated such as bandwidth sharing protocols.
- Too complicated for algorithm design.

Two considered models: good trade-off realism/tractability

Platform model: communication models

- Multi-port: if several non-consecutive stages mapped onto a same processor, several concurrent communications
- Matches multi-threaded systems
- Fits well together with overlap

- One-port: radical option, where everything is serialized
- Natural to consider it without overlap

- Other communication models: more complicated such as bandwidth sharing protocols.
- Too complicated for algorithm design.

Two considered models: good trade-off realism/tractability

Platform model: communication models

- Multi-port: if several non-consecutive stages mapped onto a same processor, several concurrent communications
- Matches multi-threaded systems
- Fits well together with overlap

- One-port: radical option, where everything is serialized
- Natural to consider it without overlap

- Other communication models: more complicated such as bandwidth sharing protocols.
- Too complicated for algorithm design.

Two considered models: good trade-off realism/tractability

Platform model: communication models

- Multi-port: if several non-consecutive stages mapped onto a same processor, several concurrent communications
- Matches multi-threaded systems
- Fits well together with overlap

- One-port: radical option, where everything is serialized
- Natural to consider it without overlap

- Other communication models: more complicated such as bandwidth sharing protocols.
- Too complicated for algorithm design.

Two considered models: good trade-off realism/tractability

Multi-criteria mapping problems

- Goal: assign application stages to platform processors in order to optimize some criteria
- Define stage types and replication mechanisms
- Establish rule of the game
- Define optimization criteria
- Define and classify optimization problems

Multi-criteria mapping problems

- Goal: assign application stages to platform processors in order to optimize some criteria
- Define stage types and replication mechanisms
- Establish rule of the game
- Define optimization criteria
- Define and classify optimization problems

Mapping: stage types and replication

- Monolithic stages: must be mapped on one single processor since computation for a data set may depend on result of previous computation
- Dealable stages: can be replicated on several processors, but not parallel, *i.e.* a data set must be entirely processed on a single processor (distribute work)
- Data-parallel stages: inherently parallel stages, one data set can be computed in parallel by several processors (partition work)

- Replicating for failures: one data set is processed several times on different processors (redundant work)

Mapping: stage types and replication

- Monolithic stages: must be mapped on one single processor since computation for a data set may depend on result of previous computation
- Dealable stages: can be replicated on several processors, but not parallel, *i.e.* a data set must be entirely processed on a single processor (distribute work)
- Data-parallel stages: inherently parallel stages, one data set can be computed in parallel by several processors (partition work)

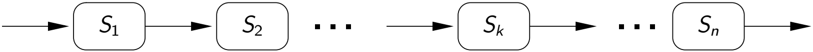
- Replicating for failures: one data set is processed several times on different processors (redundant work)

Mapping strategies: rule of the game

- Map each application stage onto one or more processors
- First simple scenario with no replication

- Allocation function $a : [1..n] \rightarrow [1..p]$
- $a(0) = 0$ $(= in)$ and $a(n+1) = p + 1$ $(= out)$

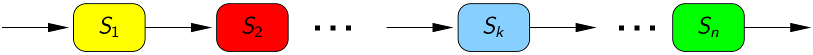
- Several mapping strategies



The pipeline application

Mapping strategies: rule of the game

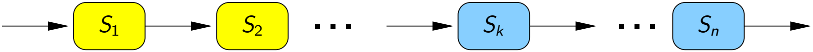
- Map each application stage onto one or more processors
- First simple scenario with no replication
- Allocation function $a : [1..n] \rightarrow [1..p]$
- $a(0) = 0$ $(= in)$ and $a(n+1) = p+1$ $(= out)$
- Several mapping strategies



ONE-TO-ONE MAPPING: a is a one-to-one function, $n \leq p$

Mapping strategies: rule of the game

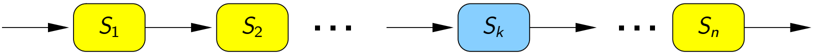
- Map each application stage onto one or more processors
- First simple scenario with no replication
- Allocation function $a : [1..n] \rightarrow [1..p]$
- $a(0) = 0$ ($= in$) and $a(n+1) = p+1$ ($= out$)
- Several mapping strategies



INTERVAL MAPPING: partition into $m \leq p$ intervals $I_j = [d_j, e_j]$

Mapping strategies: rule of the game

- Map each application stage onto one or more processors
- First simple scenario with no replication
- Allocation function $a : [1..n] \rightarrow [1..p]$
- $a(0) = 0$ $(= in)$ and $a(n+1) = p+1$ $(= out)$
- Several mapping strategies



General Mapping: P_u is assigned any subset of stages

Mapping strategies: adding replication

- Allocation function: $a(i)$ is a set of processor indices
- Set partitioned into t_i *teams*, each processor within a team is allocated the same piece of work
- Teams for stage S_i: $T_{i,1}, \ldots, T_{i,t_i}$ $(1 \leq i \leq n)$
- Monolithic stage: single team $t_i = 1$ and $|T_{i,1}| = |a(i)|$; replication only for reliability if $|a(i)| > 1$
- Dealable stage: each team = one round of the deal; $type_i = deal$
- Data-parallel stage: each team = computation of a fraction of each data set; $type_i = dp$
- Extend mapping rules with replication, same teams for an interval or a subset of stages; no fully general mappings

Mapping strategies: adding replication

- Allocation function: $a(i)$ is a set of processor indices
- Set partitioned into t_i *teams*, each processor within a team is allocated the same piece of work
- Teams for stage S_i: $T_{i,1}, \dots, T_{i,t_i}$ $(1 \leq i \leq n)$
- Monolithic stage: single team $t_i = 1$ and $|T_{i,1}| = |a(i)|$; replication only for reliability if $|a(i)| > 1$
- Dealable stage: each team = one round of the deal; $type_i = deal$
- Data-parallel stage: each team = computation of a fraction of each data set; $type_i = dp$
- Extend mapping rules with replication, same teams for an interval or a subset of stages; no fully general mappings

Mapping strategies: adding replication

- Allocation function: $a(i)$ is a set of processor indices
- Set partitioned into t_i *teams*, each processor within a team is allocated the same piece of work
- Teams for stage S_i: $T_{i,1}, \ldots, T_{i,t_i}$ $(1 \leq i \leq n)$

- Monolithic stage: single team $t_i = 1$ and $|T_{i,1}| = |a(i)|$; replication only for reliability if $|a(i)| > 1$
- Dealable stage: each team = one round of the deal; $type_i = deal$
- Data-parallel stage: each team = computation of a fraction of each data set; $type_i = dp$

- Extend mapping rules with replication, same teams for an interval or a subset of stages; no fully general mappings

Mapping: objective function

Mono-criterion

- Minimize period $\mathcal{P}$ (inverse of throughput)
- Minimize latency $\mathcal{L}$ (time to process a data set)
- Minimize application failure probability $\mathcal{F}$

Mapping: objective function

Mono-criterion

- Minimize period $\mathcal{P}$ (inverse of throughput)
- Minimize latency $\mathcal{L}$ (time to process a data set)
- Minimize application failure probability $\mathcal{F}$

Multi-criteria

- How to define it?
  Minimize $\alpha.\mathcal{P} + \beta.\mathcal{L} + \gamma.\mathcal{F}$?
- Values which are not comparable

Mapping: objective function

Mono-criterion

- Minimize period $\mathcal{P}$ (inverse of throughput)
- Minimize latency $\mathcal{L}$ (time to process a data set)
- Minimize application failure probability $\mathcal{F}$

Multi-criteria

- How to define it?
  Minimize $\alpha.\mathcal{P} + \beta.\mathcal{L} + \gamma.\mathcal{F}$?
- Values which are not comparable

- Minimize $\mathcal{P}$ for a fixed latency and failure
- Minimize $\mathcal{L}$ for a fixed period and failure
- Minimize $\mathcal{F}$ for a fixed period and latency

Mapping: objective function

Mono-criterion

- Minimize period $\mathcal{P}$ (inverse of throughput)
- Minimize latency $\mathcal{L}$ (time to process a data set)
- Minimize application failure probability $\mathcal{F}$

Bi-criteria

- Period and Latency:
- Minimize $\mathcal{P}$ for a fixed latency
- Minimize $\mathcal{L}$ for a fixed period
- And so on...

Formal definition of period and latency

- Allocation function: characterizes a mapping
- Not enough information to compute the actual schedule of the application = the moment at which each operation takes place
- Time steps at which comm and comp begin and end
- Cyclic schedules which repeat for each data set (period λ)

- No deal replication: S_i, $u \in a(i)$, $v \in a(i+1)$, data set k
  - $BeginComp^k_{i,u}/EndComp^k_{i,u}$ = time step at which comp of S_i on P_u for data set k begins/ends
  - $BeginComm^k_{i,u,v}/EndComm^k_{i,u,v}$ = time step at which comm between P_u and P_v for output of S_i for k begins/ends

$$\begin{cases} BeginComp^k_{i,u} = BeginComp^0_{i,u} + \lambda \times k \\ EndComp^k_{i,u} = EndComp^0_{i,u} + \lambda \times k \\ BeginComm^k_{i,u,v} = BeginComm^0_{i,u,v} + \lambda \times k \\ EndComm^k_{i,u,v} = EndComm^0_{i,u,v} + \lambda \times k \end{cases}$$

Formal definition of period and latency

- Allocation function: characterizes a mapping
- Not enough information to compute the actual schedule of the application = the moment at which each operation takes place
- Time steps at which comm and comp begin and end
- Cyclic schedules which repeat for each data set (period λ)

- No deal replication: S_i, $u \in a(i)$, $v \in a(i+1)$, data set k
  - $BeginComp^k_{i,u}/EndComp^k_{i,u}$ = time step at which comp of S_i on P_u for data set k begins/ends
  - $BeginComm^k_{i,u,v}/EndComm^k_{i,u,v}$ = time step at which comm between P_u and P_v for output of S_i for k begins/ends

$$\begin{cases} BeginComp^k_{i,u} = BeginComp^0_{i,u} + \lambda \times k \\ EndComp^k_{i,u} = EndComp^0_{i,u} + \lambda \times k \\ BeginComm^k_{i,u,v} = BeginComm^0_{i,u,v} + \lambda \times k \\ EndComm^k_{i,u,v} = EndComm^0_{i,u,v} + \lambda \times k \end{cases}$$

Formal definition of period and latency

- Allocation function: characterizes a mapping
- Not enough information to compute the actual schedule of the application = the moment at which each operation takes place
- Time steps at which comm and comp begin and end
- Cyclic schedules which repeat for each data set (period λ)

- No deal replication: S_i, $u \in a(i)$, $v \in a(i+1)$, data set k
  - $BeginComp^k_{i,u} / EndComp^k_{i,u}$ = time step at which comp of S_i on P_u for data set k begins/ends
  - $BeginComm^k_{i,u,v} / EndComm^k_{i,u,v}$ = time step at which comm between P_u and P_v for output of S_i for k begins/ends

$$\begin{cases} BeginComp^k_{i,u} = BeginComp^0_{i,u} + \lambda \times k \\ EndComp^k_{i,u} = EndComp^0_{i,u} + \lambda \times k \\ BeginComm^k_{i,u,v} = BeginComm^0_{i,u,v} + \lambda \times k \\ EndComm^k_{i,u,v} = EndComm^0_{i,u,v} + \lambda \times k \end{cases}$$

Formal definition of period and latency: *operation list*

- Given communication model: set of rules to have a valid operation list
- Non-preemptive models, synchronous communications
- Period $\mathcal{P} = \lambda$
- Latency $\mathcal{L} = \max\{EndComm^0_{n,u,out} \mid u \in a(n),\}$
- With deal replication: extension of the definition, periodic schedule rather than cyclic one
- Most cases: formula to express period and latency, no need for OL

Now, ready to describe optimization problems

Formal definition of period and latency: *operation list*

- Given communication model: set of rules to have a valid operation list
- Non-preemptive models, synchronous communications
- Period $\mathcal{P} = \lambda$
- Latency $\mathcal{L} = \max\{EndComm^{0}_{n,u,out} \mid u \in a(n),\}$
- With deal replication: extension of the definition, periodic schedule rather than cyclic one
- Most cases: formula to express period and latency, no need for OL

Now, ready to describe optimization problems

Formal definition of period and latency: *operation list*

- Given communication model: set of rules to have a valid operation list
- Non-preemptive models, synchronous communications
- Period $\mathcal{P} = \lambda$
- Latency $\mathcal{L} = \max\{EndComm^{0}_{n,u,out} \mid u \in a(n),\}$
- With deal replication: extension of the definition, periodic schedule rather than cyclic one
- Most cases: formula to express period and latency, no need for OL

Now, ready to describe optimization problems

Formal definition of period and latency: *operation list*

- Given communication model: set of rules to have a valid operation list
- Non-preemptive models, synchronous communications
- Period $\mathcal{P} = \lambda$
- Latency $\mathcal{L} = \max\{EndComm^{0}_{n,u,out} \mid u \in a(n), \}$
- With deal replication: extension of the definition, periodic schedule rather than cyclic one
- Most cases: formula to express period and latency, no need for OL

Now, ready to describe optimization problems

Formal definition of period and latency: *operation list*

- Given communication model: set of rules to have a valid operation list
- Non-preemptive models, synchronous communications
- Period $\mathcal{P} = \lambda$
- Latency $\mathcal{L} = \max\{EndComm^{0}_{n,u,out} \mid u \in a(n), \}$
- With deal replication: extension of the definition, periodic schedule rather than cyclic one
- Most cases: formula to express period and latency, no need for OL

Now, ready to describe optimization problems

One-to-one and interval mappings, no replication

- Latency: max time required by a data set to traverse all stages

$$\mathcal{L}^{(interval)} = \sum_{1 \le j \le m} \left\{ \frac{\delta_{d_j - 1}}{b_{a(d_j-1),a(d_j)}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{a(d_j)}} \right\} + \frac{\delta_n}{b_{a(d_m),out}}$$

- Period: definition depends on comm model (different rules in the OL), but always longest cycle-time of a processor:
  $\mathcal{P}^{(interval)} = \max_{1 \le j \le m} cycletime(P_{a(d_j)})$
- One-port model without overlap:

$$\mathcal{P} = \max_{1 \le j \le m} \left\{ \frac{\delta_{d_j - 1}}{b_{a(d_j-1),a(d_j)}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{a(d_j)}} + \frac{\delta_{e_j}}{b_{a(d_j),a(e_j+1)}} \right\}$$

- Bounded multi-port model with overlap:

One-to-one and interval mappings, no replication

- Latency: max time required by a data set to traverse all stages

$$\mathcal{L}^{(interval)} = \sum_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{b_{a(d_j-1),a(d_j)}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{a(d_j)}} \right\} + \frac{\delta_n}{b_{a(d_m),out}}$$

- Period: definition depends on comm model (different rules in the OL), but always longest cycle-time of a processor: $\mathcal{P}^{(interval)} = \max_{1 \leq j \leq m} cycletime(P_{a(d_j)})$

- One-port model without overlap:

$$\mathcal{P} = \max_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{b_{a(d_j-1),a(d_j)}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{a(d_j)}} + \frac{\delta_{e_j}}{b_{a(d_j),a(e_j+1)}} \right\}$$

- Bounded multi-port model with overlap:

One-to-one and interval mappings, no replication

- Latency: max time required by a data set to traverse all stages

$$\mathcal{L}^{(interval)} = \sum_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{b_{a(d_j-1),a(d_j)}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{a(d_j)}} \right\} + \frac{\delta_n}{b_{a(d_m),out}}$$

- Period: definition depends on comm model (different rules in the OL), but always longest cycle-time of a processor:
  $\mathcal{P}^{(interval)} = \max_{1 \leq j \leq m} cycletime(P_{a(d_j)})$

- One-port model without overlap:

$$\mathcal{P} = \max_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{b_{a(d_j-1),a(d_j)}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{a(d_j)}} + \frac{\delta_{e_j}}{b_{a(d_j),a(e_j+1)}} \right\}$$

- Bounded multi-port model with overlap:

One-to-one and interval mappings, no replication

- Latency: max time required by a data set to traverse all stages

$$\mathcal{L}^{(interval)} = \sum_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{b_{a(d_j - 1), a(d_j)}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{a(d_j)}} \right\} + \frac{\delta_n}{b_{a(d_m), out}}$$

- Period: definition depends on comm model (different rules in the OL), but always longest cycle-time of a processor:
  $\mathcal{P}^{(interval)} = \max_{1 \leq j \leq m} cycletime(P_{a(d_j)})$
- One-port model without overlap:

$$\mathcal{P} = \max_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{b_{a(d_j - 1), a(d_j)}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{a(d_j)}} + \frac{\delta_{e_j}}{b_{a(d_j), a(e_j + 1)}} \right\}$$

- Bounded multi-port model with overlap:

$$\mathcal{P} = \max_{1 \leq j \leq m} \left\{ max\left(\frac{\delta_{d_j - 1}}{\min\left(b_{a(d_j - 1), a(d_j)}, \mathrm{B}^i_{a(d_j)} \right)}, \frac{\sum_{i=d_j}^{e_j} w_i}{s_{a(d_j)}}, \frac{\delta_{e_j}}{\min\left(b_{a(d_j), a(e_j + 1)}, \mathrm{B}^o_{a(d_j)} \right)} \right) \right\}$$

Adding replication for reliability

- Each processor: failure probability $0 \leq f_u \leq 1$
- m intervals, set of processors $a(d_j)$ for interval j

$$\mathcal{F}^{(int-fp)} = 1 - \prod_{1 \leq j \leq m} \left(1 - \prod_{u \in a(d_j)} f_u\right)$$

- Consensus protocol: one surviving processor performs all outgoing communications
- Worst case scenario: new formulas for latency and period

$$\mathcal{L}^{(int-fp)} = \sum_{u \in a(1)} \frac{\delta_0}{b_{in,u}} + \sum_{1 \leq j \leq m} \max_{u \in a(d_j)} \left\{ \frac{\sum_{i=d_j}^{e_j} w_i}{s_u} + \sum_{v \in a(e_j+1)} \frac{\delta_{e_j}}{b_{u,v}} \right\}$$

$$\mathcal{P}^{(int-fp)} = \max_{1 \leq j \leq m} \max_{u \in a(d_j)} \left\{ \frac{\delta_{d_j-1}}{\min_{v \in a(d_j-1)} b_{v,u}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_u} + \sum_{v \in a(e_j+1)} \frac{\delta_{e_j}}{b_{u,v}} \right\}$$

Adding replication for reliability

- Each processor: failure probability $0 \leq \mathrm{f}_u \leq 1$
- m intervals, set of processors $a(d_j)$ for interval j

$$\mathcal{F}^{(int-fp)} = 1 - \prod_{1 \leq j \leq m} \left(1 - \prod_{u \in a(d_j)} \mathrm{f}_u\right)$$

- Consensus protocol: one surviving processor performs all outgoing communications
- Worst case scenario: new formulas for latency and period

$$\mathcal{L}^{(int-fp)} = \sum_{u \in a(1)} \frac{\delta_0}{b_{in,u}} + \sum_{1 \leq j \leq m} \max_{u \in a(d_j)} \left\{ \frac{\sum_{i=d_j}^{e_j} w_i}{s_u} + \sum_{v \in a(e_j+1)} \frac{\delta_{e_j}}{b_{u,v}} \right\}$$

$$\mathcal{P}^{(int-fp)} = \max_{1 \leq j \leq m} \max_{u \in a(d_j)} \left\{ \frac{\delta_{d_j-1}}{\min_{v \in a(d_j-1)} b_{v,u}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_u} + \sum_{v \in a(e_j+1)} \frac{\delta_{e_j}}{b_{u,v}} \right\}$$

Adding replication for reliability

- Each processor: failure probability $0 \leq \mathrm{f}_u \leq 1$
- m intervals, set of processors $a(d_j)$ for interval j

$$\mathcal{F}^{(int-fp)} = 1 - \prod_{1 \leq j \leq m} \left(1 - \prod_{u \in a(d_j)} \mathrm{f}_u\right)$$

- Consensus protocol: one surviving processor performs all outgoing communications
- Worst case scenario: new formulas for latency and period

$$\mathcal{L}^{(int-fp)} = \sum_{u \in a(1)} \frac{\delta_0}{b_{in,u}} + \sum_{1 \leq j \leq m} \max_{u \in a(d_j)} \left\{ \frac{\sum_{i=d_j}^{e_j} w_i}{s_u} + \sum_{v \in a(e_j+1)} \frac{\delta_{e_j}}{b_{u,v}} \right\}$$

$$\mathcal{P}^{(int-fp)} = \max_{1 \leq j \leq m} \max_{u \in a(d_j)} \left\{ \frac{\delta_{d_j-1}}{\min_{v \in a(d_j-1)} b_{v,u}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_u} + \sum_{v \in a(e_j+1)} \frac{\delta_{e_j}}{b_{u,v}} \right\}$$

Adding replication for reliability

- Each processor: failure probability $0 \leq f_u \leq 1$
- m intervals, set of processors $a(d_j)$ for interval j

$$\mathcal{F}^{(int-fp)} = 1 - \prod_{1 \leq j \leq m} \left(1 - \prod_{u \in a(d_j)} f_u\right)$$

- Consensus protocol: one surviving processor performs all outgoing communications
- Worst case scenario: new formulas for latency and period

$$\mathcal{L}^{(int-fp)} = \sum_{u \in a(1)} \frac{\delta_0}{b_{in,u}} + \sum_{1 \leq j \leq m} \max_{u \in a(d_j)} \left\{ \frac{\sum_{i=d_j}^{e_j} w_i}{s_u} + \sum_{v \in a(e_j+1)} \frac{\delta_{e_j}}{b_{u,v}} \right\}$$

$$\mathcal{P}^{(int-fp)} = \max_{1 \leq j \leq m} \max_{u \in a(d_j)} \left\{ \frac{\delta_{d_j-1}}{\min_{v \in a(d_j-1)} b_{v,u}} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_u} + \sum_{v \in a(e_j+1)} \frac{\delta_{e_j}}{b_{u,v}} \right\}$$

Adding replication for period and latency

- Dealable stages: replication of stage or interval of stages.
  - No latency decrease; period may decrease (less data sets per processor)
  - No communication: period $trav_i/k$ if S_i onto k processors; $trav_i = \frac{w_i}{min_{1 \leq u \leq k} s_{q_u}}$
  - With communications: cases with no critical resources
  - Latency: longest path, no conflicts between data sets
- Data-parallel stages: replication of single stage
  - Both latency and period may decrease
  - $trav_i = o_i + \frac{w_i}{\sum_{u=1}^{k} s_{q_u}}$
  - Becomes very difficult with communications
- $\Rightarrow$ Model with no communication!
- Replication for performance + replication for reliability: possible to mix both approaches, difficulties of both models

Adding replication for period and latency

- Dealable stages: replication of stage or interval of stages.
  - No latency decrease; period may decrease (less data sets per processor)
  - No communication: period $trav_i/k$ if S_i onto k processors; $trav_i = \frac{w_i}{min_{1 \leq u \leq k} s_{q_u}}$
  - With communications: cases with no critical resources
  - Latency: longest path, no conflicts between data sets
- Data-parallel stages: replication of single stage
  - Both latency and period may decrease
  - $trav_i = o_i + \frac{w_i}{\sum_{u=1}^{k} s_{q_u}}$
  - Becomes very difficult with communications
- $\Rightarrow$ Model with no communication!
- Replication for performance + replication for reliability: possible to mix both approaches, difficulties of both models

Adding replication for period and latency

- Dealable stages: replication of stage or interval of stages.
  - No latency decrease; period may decrease (less data sets per processor)
  - No communication: period $trav_i/k$ if S_i onto k processors; $trav_i = \frac{w_i}{min_{1 \leq u \leq k} s_{q_u}}$
  - With communications: cases with no critical resources
  - Latency: longest path, no conflicts between data sets
- Data-parallel stages: replication of single stage
  - Both latency and period may decrease
  - $trav_i = o_i + \frac{w_i}{\sum_{u=1}^{k} s_{q_u}}$
  - Becomes very difficult with communications
- $\Rightarrow$ Model with no communication!
- Replication for performance + replication for reliability: possible to mix both approaches, difficulties of both models

Adding replication for period and latency

- Dealable stages: replication of stage or interval of stages.
  - No latency decrease; period may decrease (less data sets per processor)
  - No communication: period $trav_i/k$ if S_i onto k processors; $trav_i = \frac{w_i}{min_{1 \leq u \leq k} s_{q_u}}$
  - With communications: cases with no critical resources
  - Latency: longest path, no conflicts between data sets
- Data-parallel stages: replication of single stage
  - Both latency and period may decrease
  - $trav_i = o_i + \frac{w_i}{\sum_{u=1}^{k} s_{q_u}}$
  - Becomes very difficult with communications
- $\Rightarrow$ Model with no communication!
- Replication for performance + replication for reliability: possible to mix both approaches, difficulties of both models

Moving to general mappings

- Failure probability: definition in the general case easy to derive (all kind of replication)

$$\mathcal{F}^{(gen)} = 1 - \prod_{1 \leq j \leq m} \prod_{1 \leq k \leq t_{d_j}} \left(1 - \prod_{u \in T_{d_j,k}} \mathsf{f}_u\right)$$

- Latency: can be defined for *Communication Homogeneous* platforms with no data-parallelism.

$$\mathcal{L}^{(gen)} = \sum_{1 \leq i \leq n} \left(\max_{1 \leq k \leq t_i} \left\{ \Delta_i |T_{i,k}| \frac{\delta_{i-1}}{b} + \frac{w_i}{\min_{u \in T_{i,k}} s_u} \right\} \right) + \frac{\delta_{n+1}}{b}$$

  - $\Delta_i = 1$ iff S_{i-1} and S_i are in the same subset
  - *Fully Heterogeneous*: longest path computation (polynomial time)
  - With data-parallel stages: can be computed only with no communication and no start-up overhead

Moving to general mappings

- Failure probability: definition in the general case easy to derive (all kind of replication)

$$\mathcal{F}^{(gen)} = 1 - \prod_{1 \le j \le m} \prod_{1 \le k \le t_{d_j}} \left(1 - \prod_{u \in T_{d_j,k}} \mathsf{f}_u\right)$$

- Latency: can be defined for *Communication Homogeneous* platforms with no data-parallelism.

$$\mathcal{L}^{(gen)} = \sum_{1 \le i \le n} \left(\max_{1 \le k \le t_i} \left\{ \Delta_i |T_{i,k}| \frac{\delta_{i-1}}{b} + \frac{w_i}{\min_{u \in T_{i,k}} s_u} \right\} \right) + \frac{\delta_{n+1}}{b}$$

  - $\Delta_i = 1$ iff S_{i-1} and S_i are in the same subset
  - *Fully Heterogeneous*: longest path computation (polynomial time)
  - With data-parallel stages: can be computed only with no communication and no start-up overhead

Moving to general mappings

- Period: case with no replication for period and latency
- Bounded multi-port model with overlap
  - Period = maximum cycle-time of processors
  - Communications in parallel: No conflicts
    input coms on data sets $k_1+1, \dots, k_\ell+1$; computes on $k_1, \dots, k_\ell$, outputs $k_1-1, \dots, k_\ell-1$

$$\mathcal{P}^{(gen-mp)} = \max_{1 \le j \le m} \max_{u \in a(d_j)} \Bigg\{ \max \left(\max_{i \in stages_j} \max_{v \in a(i-1)} \Delta_i \frac{\delta_{i-1}}{b_{v,u}}, \sum_{i \in stages_j} \Delta_i \frac{\delta_{i-1}}{\mathrm{B}^i_u}, \frac{\sum_{i \in stages_j} w_i}{s_u}, \max_{i \in stages_j} \max_{v \in a(i+1)} \Delta_{i+1} \frac{\delta_i}{b_{u,v}}, \sum_{i \in stages_j} \Delta_{i+1} \frac{\delta_i}{\mathrm{B}^o_u} \right) \Bigg\}$$

- Without overlap: conflicts similar to case with replication; NP-hard to decide how to order coms

Moving to general mappings

- Period: case with no replication for period and latency
- Bounded multi-port model with overlap
  - Period = maximum cycle-time of processors
  - Communications in parallel: No conflicts
    input coms on data sets $k_1+1,\dots,k_\ell+1$; computes on $k_1,\dots,k_\ell$, outputs $k_1-1,\dots,k_\ell-1$

$$\mathcal{P}^{(gen-mp)} = \max_{1\leq j\leq m} \max_{u\in a(d_j)} \Bigg\{ \max \Bigg(\max_{i\in stages_j} \max_{v\in a(i-1)} \Delta_i \frac{\delta_{i-1}}{b_{v,u}}, \sum_{i\in stages_j} \Delta_i \frac{\delta_{i-1}}{\mathrm{B}^i_u}, \frac{\sum_{i\in stages_j} w_i}{s_u}, \max_{i\in stages_j} \max_{v\in a(i+1)} \Delta_{i+1} \frac{\delta_i}{b_{u,v}}, \sum_{i\in stages_j} \Delta_{i+1} \frac{\delta_i}{\mathrm{B}^o_u} \Bigg) \Bigg\}$$

- Without overlap: conflicts similar to case with replication; NP-hard to decide how to order coms

Outline

1. Models
   - Application model
   - Platform and communication models
   - Multi-criteria mapping problems

2. Complexity results
   - Mono-criterion problems
   - Bi-criteria problems

3. Conclusion

Failure probability

- Turns out simple for interval and general mappings: minimum reached by replicating the whole pipeline as a single interval consisting in a single team on all processors: $\mathcal{F} = \prod_{u=1}^{p} \mathsf{f}_u$
- One-to-one mappings: polynomial for *Failure Homogeneous* platforms (balance number of processors to stages), NP-hard for *Failure Heterogeneous* platforms (3-PARTITION with n stages and $3n$ processors)

$\mathcal{F}$	**Failure-Hom.**	**Failure-Het.**
One-to-one	polynomial	NP-hard
Interval	polynomial	
General	polynomial	

Failure probability

- Turns out simple for interval and general mappings: minimum reached by replicating the whole pipeline as a single interval consisting in a single team on all processors: $\mathcal{F} = \prod_{u=1}^{p} f_u$
- One-to-one mappings: polynomial for *Failure Homogeneous* platforms (balance number of processors to stages), NP-hard for *Failure Heterogeneous* platforms (3-PARTITION with n stages and $3n$ processors)

$\mathcal{F}$	Failure-Hom.	Failure-Het.
One-to-one	polynomial	NP-hard
Interval	polynomial	
General	polynomial	

Failure probability

- Turns out simple for interval and general mappings: minimum reached by replicating the whole pipeline as a single interval consisting in a single team on all processors: $\mathcal{F} = \prod_{u=1}^{p} \mathsf{f}_u$
- One-to-one mappings: polynomial for *Failure Homogeneous* platforms (balance number of processors to stages), NP-hard for *Failure Heterogeneous* platforms (3-PARTITION with n stages and $3n$ processors)

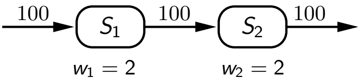
$\mathcal{F}$	**Failure-Hom.**	**Failure-Het.**
One-to-one	polynomial	NP-hard
Interval	polynomial	
General	polynomial	

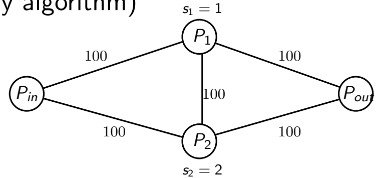
Latency

- Replication of dealable stages, replication for reliability: no impact on latency
- No data-parallelism: reduce communication costs
  - *Fully Homogeneous* and *Communication Homogeneous* platforms: map all stages onto fastest processor (1 interval); one-to-one mappings: most computationally expensive stages onto fastest processors (greedy algorithm)
  - *Fully Heterogeneous* platforms: problem of input/output communications: may need to split interval

Latency

- Replication of dealable stages, replication for reliability: no impact on latency
- No data-parallelism: reduce communication costs
  - *Fully Homogeneous* and *Communication Homogeneous* platforms: map all stages onto fastest processor (1 interval); one-to-one mappings: most computationally expensive stages onto fastest processors (greedy algorithm)

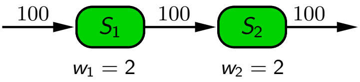


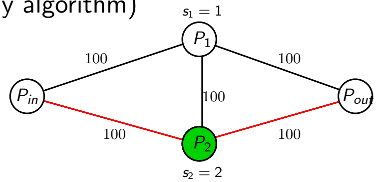


  - *Fully Heterogeneous* platforms: problem of input/output communications: may need to split interval

Latency

- Replication of dealable stages, replication for reliability: no impact on latency
- No data-parallelism: reduce communication costs
  - *Fully Homogeneous* and *Communication Homogeneous* platforms: map all stages onto fastest processor (1 interval); one-to-one mappings: most computationally expensive stages onto fastest processors (greedy algorithm)

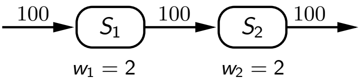


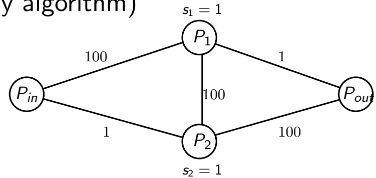


  - *Fully Heterogeneous* platforms: problem of input/output communications: may need to split interval

Latency

- Replication of dealable stages, replication for reliability: no impact on latency
- No data-parallelism: reduce communication costs
  - *Fully Homogeneous* and *Communication Homogeneous* platforms: map all stages onto fastest processor (1 interval); one-to-one mappings: most computationally expensive stages onto fastest processors (greedy algorithm)

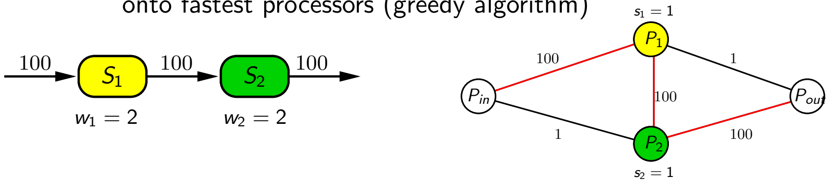




  - *Fully Heterogeneous* platforms: problem of input/output communications: may need to split interval

Latency

- Replication of dealable stages, replication for reliability: no impact on latency
- No data-parallelism: reduce communication costs
  - *Fully Homogeneous* and *Communication Homogeneous* platforms: map all stages onto fastest processor (1 interval); one-to-one mappings: most computationally expensive stages onto fastest processors (greedy algorithm)



  - *Fully Heterogeneous* platforms: problem of input/output communications: may need to split interval

Latency

- *Fully Heterogeneous* platforms: NP-hard for one-to-one and interval mappings (involved reductions), polynomial for general mappings (shortest paths)
- With data-parallelism: model with no communication; polynomial with same speed processors (dynamic programming algorithm), NP-hard otherwise (2-PARTITION)

$\mathcal{L}$	Fully Hom.	Comm. Hom.	Hetero.
no DP, One-to-one	polynomial		NP-hard
no DP, Interval	polynomial		NP-hard
no DP, General	polynomial		
with DP, no coms	polynomial	NP-hard	

Latency

- *Fully Heterogeneous* platforms: NP-hard for one-to-one and interval mappings (involved reductions), polynomial for general mappings (shortest paths)
- With data-parallelism: model with no communication; polynomial with same speed processors (dynamic programming algorithm), NP-hard otherwise (2-PARTITION)

$\mathcal{L}$	Fully Hom.	Comm. Hom.	Hetero.
no DP, One-to-one	polynomial		NP-hard
no DP, Interval	polynomial		NP-hard
no DP, General	polynomial		
with DP, no coms	polynomial	NP-hard	

Latency

- *Fully Heterogeneous* platforms: NP-hard for one-to-one and interval mappings (involved reductions), polynomial for general mappings (shortest paths)
- With data-parallelism: model with no communication; polynomial with same speed processors (dynamic programming algorithm), NP-hard otherwise (2-PARTITION)

$\mathcal{L}$	**Fully Hom.**	**Comm. Hom.**	**Hetero.**
no DP, One-to-one	polynomial		NP-hard
no DP, Interval	polynomial		NP-hard
no DP, General	polynomial		
with DP, no coms	polynomial	NP-hard	

Period - Example with no comm, no replication

$$\begin{array}{ccccccc} \mathcal{S}_1 & \longrightarrow & \mathcal{S}_2 & \longrightarrow & \mathcal{S}_3 & \longrightarrow & \mathcal{S}_4 \\ 2 & & 1 & & 3 & & 4 \end{array}$$

2 processors (P_1 and P_2) of speed 1

Optimal period?

Period - Example with no comm, no replication

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 2 & & 1 & & 3 & & 4 \end{array}$$

2 processors (P_1 and P_2) of speed 1

Optimal period?

$\mathcal{P} = 5, \quad \mathcal{S}_1\mathcal{S}_3 \rightarrow P_1, \, \mathcal{S}_2\mathcal{S}_4 \rightarrow P_2$

Perfect load-balancing in this case, but NP-hard (2-PARTITION)

Interval mapping?

Period - Example with no comm, no replication

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 2 & & 1 & & 3 & & 4 \end{array}$$

2 processors (P_1 and P_2) of speed 1

Optimal period?

$\mathcal{P} = 5, \quad \mathcal{S}_1\mathcal{S}_3 \rightarrow P_1, \mathcal{S}_2\mathcal{S}_4 \rightarrow P_2$

Perfect load-balancing in this case, but NP-hard (2-PARTITION)

Interval mapping?

$\mathcal{P} = 6, \quad \mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1, \mathcal{S}_4 \rightarrow P_2$ – Polynomial algorithm?

Period - Example with no comm, no replication

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 2 & & 1 & & 3 & & 4 \end{array}$$

2 processors (P_1 and P_2) of speed 1

Optimal period?

$\mathcal{P} = 5$, $\mathcal{S}_1\mathcal{S}_3 \rightarrow P_1$, $\mathcal{S}_2\mathcal{S}_4 \rightarrow P_2$

Perfect load-balancing in this case, but NP-hard (2-PARTITION)

Interval mapping?

$\mathcal{P} = 6$, $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1$, $\mathcal{S}_4 \rightarrow P_2$ – Polynomial algorithm?

Classical chains-on-chains problem, dynamic programming works

Period - Example with no comm, no replication

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 2 & & 1 & & 3 & & 4 \end{array}$$

P_1 of speed 2, and P_2 of speed 3

Optimal period?

$\mathcal{P} = 5, \quad \mathcal{S}_1\mathcal{S}_3 \rightarrow P_1, \mathcal{S}_2\mathcal{S}_4 \rightarrow P_2$

Perfect load-balancing in this case, but NP-hard (2-PARTITION)

Interval mapping?

$\mathcal{P} = 6, \quad \mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1, \mathcal{S}_4 \rightarrow P_2$ – Polynomial algorithm?

Classical chains-on-chains problem, dynamic programming works

Heterogeneous platform?

Period - Example with no comm, no replication

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 2 & & 1 & & 3 & & 4 \end{array}$$

P_1 of speed 2, and P_2 of speed 3

Optimal period?

$\mathcal{P} = 5, \quad \mathcal{S}_1\mathcal{S}_3 \rightarrow P_1, \mathcal{S}_2\mathcal{S}_4 \rightarrow P_2$

Perfect load-balancing in this case, but NP-hard (2-PARTITION)

Interval mapping?

$\mathcal{P} = 6, \quad \mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1, \mathcal{S}_4 \rightarrow P_2$ – Polynomial algorithm?

Classical chains-on-chains problem, dynamic programming works

Heterogeneous platform?

$\mathcal{P} = 2, \quad \mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_2, \mathcal{S}_4 \rightarrow P_1$

Heterogeneous chains-on-chains, NP-hard

Period - Complexity

$\mathcal{P}$	**Fully Hom.**	**Comm. Hom.**	**Hetero.**
One-to-one	polynomial	polynomial	NP-hard
Interval	polynomial	NP-hard	NP-hard
General	NP-hard	NP-hard	

- With replication?
  - No change in complexity except one-to-one/com-hom (the problem becomes NP-hard, reduction from 2-PARTITION, enforcing use of data-parallelism) and general/full-hom (the problem becomes polynomial)
  - Other NP-completeness proofs remain valid
  - Fully homogeneous platforms: one interval replicated onto all processors (works also for general mappings); greedy assignment for one-to-one mappings

Period - Complexity

$\mathcal{P}$	Fully Hom.	Comm. Hom.	Hetero.
One-to-one	polynomial	polynomial	NP-hard
Interval	polynomial	NP-hard	NP-hard
General	NP-hard	NP-hard	

- With replication?
  - No change in complexity except one-to-one/com-hom (the problem becomes NP-hard, reduction from 2-PARTITION, enforcing use of data-parallelism) and general/full-hom (the problem becomes polynomial)
  - Other NP-completeness proofs remain valid
  - Fully homogeneous platforms: one interval replicated onto all processors (works also for general mappings); greedy assignment for one-to-one mappings

Period - Complexity

$\mathcal{P}$	**Fully Hom.**	**Comm. Hom.**	**Hetero.**
One-to-one	polynomial	polynomial, NP-hard (rep)	NP-hard
Interval	polynomial	NP-hard	NP-hard
General	NP-hard, poly (rep)	NP-hard	

- With replication?
  - No change in complexity except one-to-one/com-hom (the problem becomes NP-hard, reduction from 2-PARTITION, enforcing use of data-parallelism) and general/full-hom (the problem becomes polynomial)
  - Other NP-completeness proofs remain valid
  - Fully homogeneous platforms: one interval replicated onto all processors (works also for general mappings); greedy assignment for one-to-one mappings

Impact of communication models

$$\xrightarrow{1} \underset{2}{\mathcal{S}_1} \xrightarrow{4} \underset{1}{\mathcal{S}_2} \xrightarrow{4} \underset{3}{\mathcal{S}_3} \xrightarrow{1} \underset{4}{\mathcal{S}_4} \xrightarrow{1}$$

2 processors of speed 1

Without overlap: optimal period and latency?

General mappings: too difficult to handle in this case (no formula for latency and period) $\rightarrow$ restrict to interval mappings

$\mathcal{P} = 8$: $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1$, $\mathcal{S}_4 \rightarrow P_2$

$\mathcal{L} = 12$: $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3\mathcal{S}_4 \rightarrow P_1$

Impact of communication models

$$\xrightarrow{1} \underset{2}{\mathcal{S}_1} \xrightarrow{4} \underset{1}{\mathcal{S}_2} \xrightarrow{4} \underset{3}{\mathcal{S}_3} \xrightarrow{1} \underset{4}{\mathcal{S}_4} \xrightarrow{1}$$

2 processors of speed 1

Without overlap: optimal period and latency?

General mappings: too difficult to handle in this case (no formula for latency and period) $\rightarrow$ restrict to interval mappings

$\mathcal{P} = 8$: $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1$, $\mathcal{S}_4 \rightarrow P_2$

$\mathcal{L} = 12$: $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3\mathcal{S}_4 \rightarrow P_1$

Impact of communication models

$$\xrightarrow{1} \underset{2}{\mathcal{S}_1} \xrightarrow{4} \underset{1}{\mathcal{S}_2} \xrightarrow{4} \underset{3}{\mathcal{S}_3} \xrightarrow{1} \underset{4}{\mathcal{S}_4} \xrightarrow{1}$$

2 processors of speed 1

Without overlap: optimal period and latency?

General mappings: too difficult to handle in this case (no formula for latency and period) $\rightarrow$ restrict to interval mappings

$\mathcal{P} = 8$: $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1$, $\mathcal{S}_4 \rightarrow P_2$

$\mathcal{L} = 12$: $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3\mathcal{S}_4 \rightarrow P_1$

Impact of communication models

$$\xrightarrow{1} \underset{2}{\mathcal{S}_1} \xrightarrow{4} \underset{1}{\mathcal{S}_2} \xrightarrow{4} \underset{3}{\mathcal{S}_3} \xrightarrow{1} \underset{4}{\mathcal{S}_4} \xrightarrow{1}$$

2 processors of speed 1

Without overlap: optimal period and latency?

General mappings: too difficult to handle in this case (no formula for latency and period) → restrict to interval mappings

$\mathcal{P} = 8$: $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \to P_1$, $\mathcal{S}_4 \to P_2$

$\mathcal{L} = 12$: $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3\mathcal{S}_4 \to P_1$

Impact of communication models

$$\xrightarrow{1} \underset{2}{\mathcal{S}_1} \xrightarrow{4} \underset{1}{\mathcal{S}_2} \xrightarrow{4} \underset{3}{\mathcal{S}_3} \xrightarrow{1} \underset{4}{\mathcal{S}_4} \xrightarrow{1}$$

2 processors of speed 1

Without overlap: optimal period and latency?

General mappings: too difficult to handle in this case (no formula for latency and period) $\rightarrow$ restrict to interval mappings

$\mathcal{P} = 8$: $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1$, $\mathcal{S}_4 \rightarrow P_2$

$\mathcal{L} = 12$: $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3\mathcal{S}_4 \rightarrow P_1$

Impact of communication models

$$\xrightarrow{1} \underset{2}{\mathcal{S}_1} \xrightarrow{4} \underset{1}{\mathcal{S}_2} \xrightarrow{4} \underset{3}{\mathcal{S}_3} \xrightarrow{1} \underset{4}{\mathcal{S}_4} \xrightarrow{1}$$

2 processors of speed 1

With overlap: optimal period?

Impact of communication models

$$\xrightarrow{1} \underset{2}{\mathcal{S}_1} \xrightarrow{4} \underset{1}{\mathcal{S}_2} \xrightarrow{4} \underset{3}{\mathcal{S}_3} \xrightarrow{1} \underset{4}{\mathcal{S}_4} \xrightarrow{1}$$

2 processors of speed 1

With overlap: optimal period?

$\mathcal{P} = 5, \quad \mathcal{S}_1\mathcal{S}_3 \rightarrow P_1, \, \mathcal{S}_2\mathcal{S}_4 \rightarrow P_2$

Perfect load-balancing both for computation and comm

Impact of communication models

$$\xrightarrow{1} \underset{2}{\mathcal{S}_1} \xrightarrow{4} \underset{1}{\mathcal{S}_2} \xrightarrow{4} \underset{3}{\mathcal{S}_3} \xrightarrow{1} \underset{4}{\mathcal{S}_4} \xrightarrow{1}$$

2 processors of speed 1

With overlap: optimal period?

$\mathcal{P} = 5, \quad \mathcal{S}_1\mathcal{S}_3 \to P_1, \mathcal{S}_2\mathcal{S}_4 \to P_2$

Optimal latency?

With only one processor, $\mathcal{L} = 12$

No internal communication to pay

Impact of communication models

$$\xrightarrow{1} \underset{2}{\mathcal{S}_1} \xrightarrow{4} \underset{1}{\mathcal{S}_2} \xrightarrow{4} \underset{3}{\mathcal{S}_3} \xrightarrow{1} \underset{4}{\mathcal{S}_4} \xrightarrow{1}$$

2 processors of speed 1

With overlap: optimal period?

$\mathcal{P} = 5, \quad \mathcal{S}_1\mathcal{S}_3 \rightarrow P_1,\ \mathcal{S}_2\mathcal{S}_4 \rightarrow P_2$

Optimal latency?

Same mapping as above: $\mathcal{L} = 21$ with no period constraint

$\mathcal{P} = 21$, no conflicts

$P_{in} \rightarrow P_1$	0 0 0
P_1	1 2 1 2/12 13 14
$P_1 \rightarrow P_2$	3 4 5 6 15
$P_2 \rightarrow P_1$	8 9 10 11
P_2	7 16 17 18 19
$P_2 \rightarrow P_{out}$	20

Impact of communication models

$$\xrightarrow{1} \mathcal{S}_1 \xrightarrow{4} \mathcal{S}_2 \xrightarrow{4} \mathcal{S}_3 \xrightarrow{1} \mathcal{S}_4 \xrightarrow{1}$$

$$\quad 2 \qquad\quad 1 \qquad\quad 3 \qquad\quad 4$$

2 processors of speed 1

With overlap: optimal period?

$\mathcal{P} = 5, \quad \mathcal{S}_1\mathcal{S}_3 \to P_1, \mathcal{S}_2\mathcal{S}_4 \to P_2$

Optimal latency?with $\mathcal{P} = 5$?

Progress step-by-step in the pipeline $\to$ no conflicts

$K = 4$ processor changes, $\mathcal{L} = (2K+1).\mathcal{P} = 9\mathcal{P} = 45$

	...	period k	period $k+1$	period $k+2$	...
$in \to P_1$	...	$\mathrm{ds}^{(k)}$	$\mathrm{ds}^{(k+1)}$	$\mathrm{ds}^{(k+2)}$	...
P_1	...	$\mathrm{ds}^{(k-1)}, \mathrm{ds}^{(k-5)}$	$\mathrm{ds}^{(k)}, \mathrm{ds}^{(k-4)}$	$\mathrm{ds}^{(k+1)}, \mathrm{ds}^{(k-3)}$	...
$P_1 \to P_2$	...	$\mathrm{ds}^{(k-2)}, \mathrm{ds}^{(k-6)}$	$\mathrm{ds}^{(k-1)}, \mathrm{ds}^{(k-5)}$	$\mathrm{ds}^{(k)}, \mathrm{ds}^{(k-4)}$	...
$P_2 \to P_1$	...	$\mathrm{ds}^{(k-4)}$	$\mathrm{ds}^{(k-3)}$	$\mathrm{ds}^{(k-2)}$	...
P_2	...	$\mathrm{ds}^{(k-3)}, \mathrm{ds}^{(k-7)}$	$\mathrm{ds}^{(k-2)}, \mathrm{ds}^{(k-6)}$	$\mathrm{ds}^{(k-1)}, \mathrm{ds}^{(k-5)}$	...
$P_2 \to out$	...	$\mathrm{ds}^{(k-8)}$	$\mathrm{ds}^{(k-7)}$	$\mathrm{ds}^{(k-6)}$	...

Bi-criteria period/latency

- Most problems NP-hard because of period
- Dynamic programming algorithm for fully homogeneous platforms
- Integer linear program for interval mappings, fully heterogeneous platforms, bi-criteria, without overlap
- Variables:
  - *Obj*: period or latency of the pipeline, depending on the objective function
  - $x_{i,u}$: 1 if S_i on P_u (0 otherwise)
  - $z_{i,u,v}$: 1 if S_i on P_u and S_{i+1} on P_v (0 otherwise)
  - first_u and last_u: integer denoting first and last stage assigned to P_u (to enforce interval constraints)

Bi-criteria period/latency

- Most problems NP-hard because of period
- Dynamic programming algorithm for fully homogeneous platforms
- Integer linear program for interval mappings, fully heterogeneous platforms, bi-criteria, without overlap
- Variables:
  - *Obj*: period or latency of the pipeline, depending on the objective function
  - $x_{i,u}$: 1 if S_i on P_u (0 otherwise)
  - $z_{i,u,v}$: 1 if S_i on P_u and S_{i+1} on P_v (0 otherwise)
  - first_u and last_u: integer denoting first and last stage assigned to P_u (to enforce interval constraints)

Bi-criteria period/latency

- Most problems NP-hard because of period
- Dynamic programming algorithm for fully homogeneous platforms
- Integer linear program for interval mappings, fully heterogeneous platforms, bi-criteria, without overlap
- Variables:
  - *Obj*: period or latency of the pipeline, depending on the objective function
  - $x_{i,u}$: 1 if S_i on P_u (0 otherwise)
  - $z_{i,u,v}$: 1 if $\mathcal{S}_i$ on P_u and S_{i+1} on P_v (0 otherwise)
  - first_u and last_u: integer denoting first and last stage assigned to P_u (to enforce interval constraints)

Linear program: constraints

Constraints on processors and links:

- $\forall i \in [0..\mathrm{n}+1], \quad \sum_u x_{i,u} = 1$
- $\forall i \in [0..\mathrm{n}], \quad \sum_{u,v} z_{i,u,v} = 1$
- $\forall i \in [0..\mathrm{n}], \forall u, v \in [0..\mathrm{p}+1], x_{i,u} + x_{i+1,v} \leq 1 + z_{i,u,v}$

Constraints on intervals:

- $\forall i \in [1..\mathrm{n}], \forall u \in [1..\mathrm{p}], \quad \mathrm{first}_u \leq i.x_{i,u} + \mathrm{n}.(1 - x_{i,u})$
- $\forall i \in [1..\mathrm{n}], \forall u \in [1..\mathrm{p}], \quad \mathrm{last}_u \geq i.x_{i,u}$
- $\forall i \in [1..\mathrm{n}-1], \forall u, v \in [1..\mathrm{p}], u \neq v,$
  $\mathrm{last}_u \leq i.z_{i,u,v} + \mathrm{n}.(1 - z_{i,u,v})$
- $\forall i \in [1..\mathrm{n}-1], \forall u, v \in [1..\mathrm{p}], u \neq v, \quad \mathrm{first}_v \geq (i+1).z_{i,u,v}$

Linear program: constraints

Constraints on processors and links:

- $\forall i \in [0..\mathrm{n}+1], \qquad \sum_u x_{i,u} = 1$
- $\forall i \in [0..\mathrm{n}], \qquad \sum_{u,v} z_{i,u,v} = 1$
- $\forall i \in [0..\mathrm{n}], \forall u, v \in [0..\mathrm{p}+1], x_{i,u} + x_{i+1,v} \leq 1 + z_{i,u,v}$

Constraints on intervals:

- $\forall i \in [1..\mathrm{n}], \forall u \in [1..\mathrm{p}], \qquad \mathrm{first}_u \leq i.x_{i,u} + \mathrm{n}.(1 - x_{i,u})$
- $\forall i \in [1..\mathrm{n}], \forall u \in [1..\mathrm{p}], \qquad \mathrm{last}_u \geq i.x_{i,u}$
- $\forall i \in [1..\mathrm{n}-1], \forall u, v \in [1..\mathrm{p}], u \neq v,$
  $\mathrm{last}_u \leq i.z_{i,u,v} + \mathrm{n}.(1 - z_{i,u,v})$
- $\forall i \in [1..\mathrm{n}-1], \forall u, v \in [1..\mathrm{p}], u \neq v, \quad \mathrm{first}_v \geq (i+1).z_{i,u,v}$

Linear program: constraints

$$\forall u \in [1..p], \sum_{i=1}^{n} \left\{ \left(\sum_{t \neq u} \frac{\delta_{i-1}}{b} z_{i-1,t,u} \right) + \frac{w_i}{s_u} x_{i,u} + \left(\sum_{v \neq u} \frac{\delta_i}{b} z_{i,u,v} \right) \right\} \leq \mathcal{P}$$

$$\sum_{u=1}^{p} \sum_{i=1}^{n} \left[\left(\sum_{t \neq u, t \in [0..p+1]} \frac{\delta_{i-1}}{b} z_{i-1,t,u} \right) + \frac{w_i}{s_u} x_{i,u} \right] + \left(\sum_{u \in [0..p]} \frac{\delta_n}{b} z_{n,u,out} \right) \leq \mathcal{L}$$

Min period with fixed latency

$Obj = \mathcal{P}$

$\mathcal{L}$ is fixed

Min latency with fixed period

$Obj = \mathcal{L}$

$\mathcal{P}$ is fixed

Linear program: constraints

$$\forall u \in [1..p], \sum_{i=1}^{n} \left\{ \left(\sum_{t \neq u} \frac{\delta_{i-1}}{b} z_{i-1,t,u} \right) + \frac{w_i}{s_u} x_{i,u} + \left(\sum_{v \neq u} \frac{\delta_i}{b} z_{i,u,v} \right) \right\} \leq \mathcal{P}$$

$$\sum_{u=1}^{p} \sum_{i=1}^{n} \left[\left(\sum_{t \neq u, t \in [0..p+1]} \frac{\delta_{i-1}}{b} z_{i-1,t,u} \right) + \frac{w_i}{s_u} x_{i,u} \right] + \left(\sum_{u \in [0..p]} \frac{\delta_n}{b} z_{n,u,out} \right) \leq \mathcal{L}$$

Min period with fixed latency

$$Obj = \mathcal{P}$$

$\mathcal{L}$ is fixed

Min latency with fixed period

$$Obj = \mathcal{L}$$

$\mathcal{P}$ is fixed

Other multi-criteria problems

- Latency/reliability: two “easy” instances, polynomial bi-criteria algorithms, single interval often optimal
- Reliability/period: mixes difficulties, period often NP-hard and reliability strongly non-linear
- Tri-criteria: even more difficult
- Experimental approach, design of polynomial heuristics for such difficult problem instances

Other multi-criteria problems

- Latency/reliability: two "easy" instances, polynomial bi-criteria algorithms, single interval often optimal
- Reliability/period: mixes difficulties, period often NP-hard and reliability strongly non-linear
- Tri-criteria: even more difficult
- Experimental approach, design of polynomial heuristics for such difficult problem instances

Other multi-criteria problems

- Latency/reliability: two "easy" instances, polynomial bi-criteria algorithms, single interval often optimal
- Reliability/period: mixes difficulties, period often NP-hard and reliability strongly non-linear
- Tri-criteria: even more difficult
- Experimental approach, design of polynomial heuristics for such difficult problem instances

Outline

Related work

Subhlok and Vondran– Pipeline on hom platforms: extended

Chains-to-chains– Heterogeneous, replicate/data-parallelize

Qishi Wu et al– Directed platform graphs (WAN); unbounded multi-port with overlap; mono-criterion problems

Mapping pipelined computations onto clusters and grids– DAG [Taura et al.], DataCutter [Saltz et al.]

Energy-aware mapping of pipelined computations– [Melhem et al.], three-criteria optimization

Scheduling task graphs on heterogeneous platforms– Acyclic task graphs scheduled on different speed processors [Topcuoglu et al.]. Communication contention: one-port model [Beaumont et al.]

Mapping pipelined computations onto special-purpose architectures– FPGA arrays [Fabiani et al.]. Fault-tolerance for embedded systems [Zhu et al.]

Conclusion

- Definition of the ingredients of scheduling: applications, platforms, multi-criteria mappings
- Surprisingly difficult problems: given a mapping, how to order communications to obtain the optimal period?
- Replication for performance and general mappings add one level of difficulty
- Cases in which application throughput not dictated by a critical resource

- Full mono-criterion complexity study, hints of multi-criteria complexity results, linear program formulation

Conclusion

- Definition of the ingredients of scheduling: applications, platforms, multi-criteria mappings
- Surprisingly difficult problems: given a mapping, how to order communications to obtain the optimal period?
- Replication for performance and general mappings add one level of difficulty
- Cases in which application throughput not dictated by a critical resource

- Full mono-criterion complexity study, hints of multi-criteria complexity results, linear program formulation

Extension to dynamic platforms

- How to handle uncertainties?
- Markovian-based model to compute the throughput of a given mapping with PEPA, performance evaluation process algebra (Murray Cole, Jane Hillston, Stephen Gilmore)
- More accurate capture of the behavior with non-markovian model based on timed Petri nets: identification of non-critical resource cases (Matthieu Gallet, Bruno Gaujal, YR)
- Failure probability related to time: problems become incredibly difficult (Arny Rosenberg, Frederic Vivien, YR)

Extension to dynamic platforms

- How to handle uncertainties? Next session
- Markovian-based model to compute the throughput of a given mapping with PEPA, performance evaluation process algebra (Murray Cole, Jane Hillston, Stephen Gilmore)
- More accurate capture of the behavior with non-markovian model based on timed Petri nets: identification of non-critical resource cases (Matthieu Gallet, Bruno Gaujal, YR)
- Failure probability related to time: problems become incredibly difficult (Arny Rosenberg, Frederic Vivien, YR)

Extension to dynamic platforms

- How to handle uncertainties? Next session
- Markovian-based model to compute the throughput of a given mapping with PEPA, performance evaluation process algebra (Murray Cole, Jane Hillston, Stephen Gilmore)
- More accurate capture of the behavior with non-markovian model based on timed Petri nets: identification of non-critical resource cases (Matthieu Gallet, Bruno Gaujal, YR)
- Failure probability related to time: problems become incredibly difficult (Arny Rosenberg, Frederic Vivien, YR)

Extension to dynamic platforms

- How to handle uncertainties? Next session
- Markovian-based model to compute the throughput of a given mapping with PEPA, performance evaluation process algebra (Murray Cole, Jane Hillston, Stephen Gilmore)
- More accurate capture of the behavior with non-markovian model based on timed Petri nets: identification of non-critical resource cases (Matthieu Gallet, Bruno Gaujal, YR)
- Failure probability related to time: problems become incredibly difficult (Arny Rosenberg, Frederic Vivien, YR)

Extension to dynamic platforms

- How to handle uncertainties? Next session
- Markovian-based model to compute the throughput of a given mapping with PEPA, performance evaluation process algebra (Murray Cole, Jane Hillston, Stephen Gilmore)
- More accurate capture of the behavior with non-markovian model based on timed Petri nets: identification of non-critical resource cases (Matthieu Gallet, Bruno Gaujal, YR)
- Failure probability related to time: problems become incredibly difficult (Arny Rosenberg, Frederic Vivien, YR)

Extension to more complex applications

- Web service applications with filtering property on stages: same challenges as for standard pipelined applications (Fanny Dufossé, YR)
- Results extended for fork or fork-join graphs, additional complexity for general DAGs (YR, Mourad Hakem)
- More complex problems of replica placement optimization, and in-network stream processing application (Veronika Rehn-Sonigo, YR)

Extension to more complex applications

- Web service applications with filtering property on stages: same challenges as for standard pipelined applications (Fanny Dufossé, YR) Next talk
- Results extended for fork or fork-join graphs, additional complexity for general DAGs (YR, Mourad Hakem)
- More complex problems of replica placement optimization, and in-network stream processing application (Veronika Rehn-Sonigo, YR)

Extension to more complex applications

- Web service applications with filtering property on stages: same challenges as for standard pipelined applications (Fanny Dufossé, YR) Next talk
- Results extended for fork or fork-join graphs, additional complexity for general DAGs (YR, Mourad Hakem)
- More complex problems of replica placement optimization, and in-network stream processing application (Veronika Rehn-Sonigo, YR)

Extension to more complex applications

- Web service applications with filtering property on stages: same challenges as for standard pipelined applications (Fanny Dufossé, YR) Next talk
- Results extended for fork or fork-join graphs, additional complexity for general DAGs (YR, Mourad Hakem)
- More complex problems of replica placement optimization, and in-network stream processing application (Veronika Rehn-Sonigo, YR)

Future work

- Experiments on linear chain applications: design of multi-criteria heuristics and experiments on real applications such as a pipelined-version of MPEG-4 encoder (Veronika, YR)
- Other research directions on linear chains:
  - Complexity of period and latency minimization once a mapping is given (Loic Magnan, Kunal Agrawal, YR)
  - Multi-application setting and energy minimization (Paul Renaud-Goud, YR)
  - Trade-offs between replication for reliability and deal replication (Loris Marchal, Oliver Sinnen)
- New applications: Filtering applications (Fanny Dufossé, YR), micro-factories with task failures (Alexandru Dobrila et al)

Future work

- Experiments on linear chain applications: design of multi-criteria heuristics and experiments on real applications such as a pipelined-version of MPEG-4 encoder (Veronika, YR)
- Other research directions on linear chains:
  - Complexity of period and latency minimization once a mapping is given (Loic Magnan, Kunal Agrawal, YR)
  - Multi-application setting and energy minimization (Paul Renaud-Goud, YR)
  - Trade-offs between replication for reliability and deal replication (Loris Marchal, Oliver Sinnen)
- New applications: Filtering applications (Fanny Dufossé, YR), micro-factories with task failures (Alexandru Dobrila et al)

Future work

- Experiments on linear chain applications: design of multi-criteria heuristics and experiments on real applications such as a pipelined-version of MPEG-4 encoder (Veronika, YR)
- Other research directions on linear chains:
  - Complexity of period and latency minimization once a mapping is given (Loic Magnan, Kunal Agrawal, YR)
  - Multi-application setting and energy minimization (Paul Renaud-Goud, YR)
  - Trade-offs between replication for reliability and deal replication (Loris Marchal, Oliver Sinnen)
- New applications: Filtering applications (Fanny Dufossé, YR), micro-factories with task failures (Alexandru Dobrila et al)

Future work

Dynamic platforms and variability

- Many challenges and open problems
- StochaGrid and ALEAE projects
- Adding non-determinism to the timed Petri net model
- Extend work with more sophisticated failure model to heterogeneous platforms
- Come up with a good and realistic model for platform failure and variability

Future work

Dynamic platforms and variability

- Many challenges and open problems
- StochaGrid and ALEAE projects
- Adding non-determinism to the timed Petri net model
- Extend work with more sophisticated failure model to heterogeneous platforms
- Come up with a good and realistic model for platform failure and variability

Future work

Dynamic platforms and variability

- Many challenges and open problems
- StochaGrid and ALEAE projects
- Adding non-determinism to the timed Petri net model
- Extend work with more sophisticated failure model to heterogeneous platforms
- Come up with a good and realistic model for platform failure and variability

Future work

Dynamic platforms and variability

- Many challenges and open problems
- StochaGrid and ALEAE projects
- Adding non-determinism to the timed Petri net model
- Extend work with more sophisticated failure model to heterogeneous platforms
- Come up with a good and realistic model for platform failure and variability