Dealing with Uncertainties in Grids through the Event-based Scheduling Approach

Dalibor Klusáček

Faculty of Informatics,
Masaryk University,
Brno, Czech Republic

xklusac@fi.muni.cz

Outline

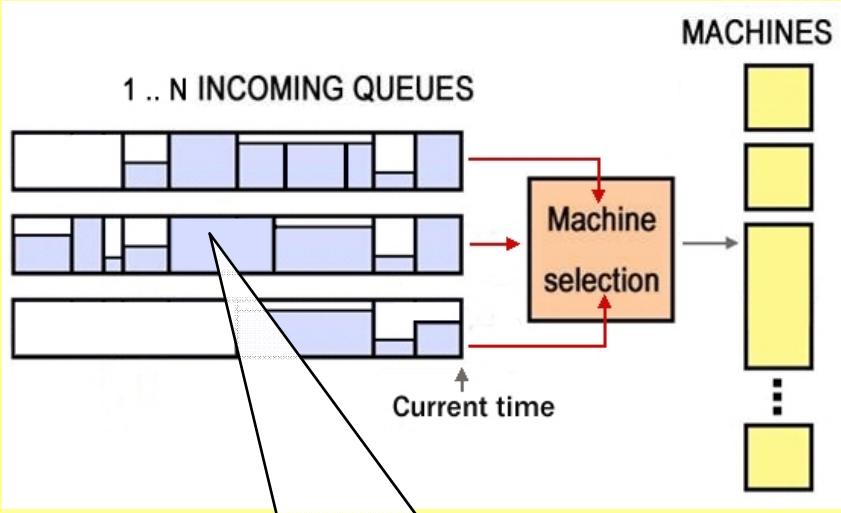
- Grid
- Grid scheduling
  - State of the Art *(Queue-based techniques)*
  - Our approach *(Schedule-based techniques)*
- Uncertainties
  - Examples
  - How they effect the schedule
- Proposed solution – event-based approach
- Preliminary Results
- Conclusion & Future Work

Introduction

- Grid
  - Large network of distributed resources
  - Dynamic, heterogeneous, decentralized
  - Transparent access, nontrivial QoS
- Grid scheduling
  - Job allocation on resources in time
  - (Complex) objective function
    - Grid owners: resource and software utilization, etc.
    - Grid users: short response time, no (few) delayed jobs, **QoS**, etc.
  - Difficult task due to dynamic behavior and uncertainty
    - The system is dynamic, heterogenous, decentralized
    - Available information is often inaccurate

State of the Art (Queue-based systems)

- 1..N queues managed by simple scheduling policies
  - Priority queues, fair queueing, round-robin, backfilling, etc.
  - Used in production systems *(PBS, LSF, MOAB, MAUI, GridWay, etc.)*

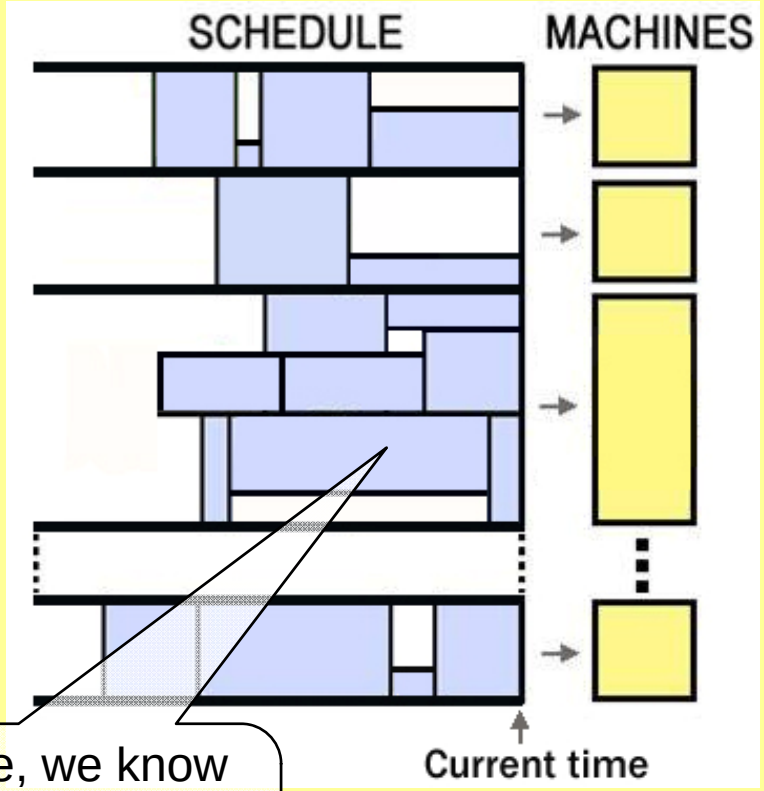


- Optimization is a hard task
  - Related with the growing amount of non-sec
  - When complex objectives involving QoS are
  - No effective support of some advanced features for general Grids
  - These techniques have no knowledge of the future Grid's behavior, i.e., what is the execution plan *(schedule)*

In the queue, the job has no idea when and where it will be executed – QoS is harder to achieve

Schedule-based Approach

- Instead of queue(s) there is a schedule
- Stores information about the future job execution, e.g.
  - Job start time *(expected)*
  - Job completion time *(expected)*
  - Selected machines *(CPUs)*
- Continuously updated due to
  - Job arrivals
  - Job completions
  - Periodical optimization
- Experimental systems
  - CCS *(Keller et al. 2001)*
  - GORBA *(Süß et al. 2005)*



Unlike to queue, we know where and when the job will be executed – QoS is easier to achieve

Uncertainties

- Examples
  - Job arrivals *(when?)*
  - Machine/network restarts & failures *(when and what?)*
  - Unprecise job execution time estimates *(10%, 50% or -70%?)*, etc.
- Related to the environment
  - Users and their jobs *(job arrivals, job description vs. real behavior)*
  - Grid itself and its dynamic behavior *(restarts & failures)*
- Uncertainties cause a lot of troubles
  - The Grid's behavior is different than expected
  - Schedule has to be modified to stay up-to-date
  - QoS is harder to achieve, etc.

Event-based Approach

- Fact: "Uncertainty and dynamic behavior is normal"
- The "state" of the Grid changes over time through events
  - Job arrivals/completions
  - Machine failures/restarts
  - Earlier/later job completions *(than was expected by the schedule)*
- These events are used as an input to the scheduler
  - Scheduler analyzes the type of the event
  - Scheduler tooks a proper action to come up with a new solution
- Two major goals
  - Keep the **QoS and other goals** (e.g., machine utilisation) over time
  - Keep the necessary **algorithm runtime low**

Supporting Techniques

- Incremental approach
  - No recomputing from scratch
  - Helps to keep the runtime reasonable
  - Reuses existing solution when building a new one
  - Introduces local changes to the schedule – preserves previous work

Typically used when:
New job arrives
Machine fails down
During optimization phase

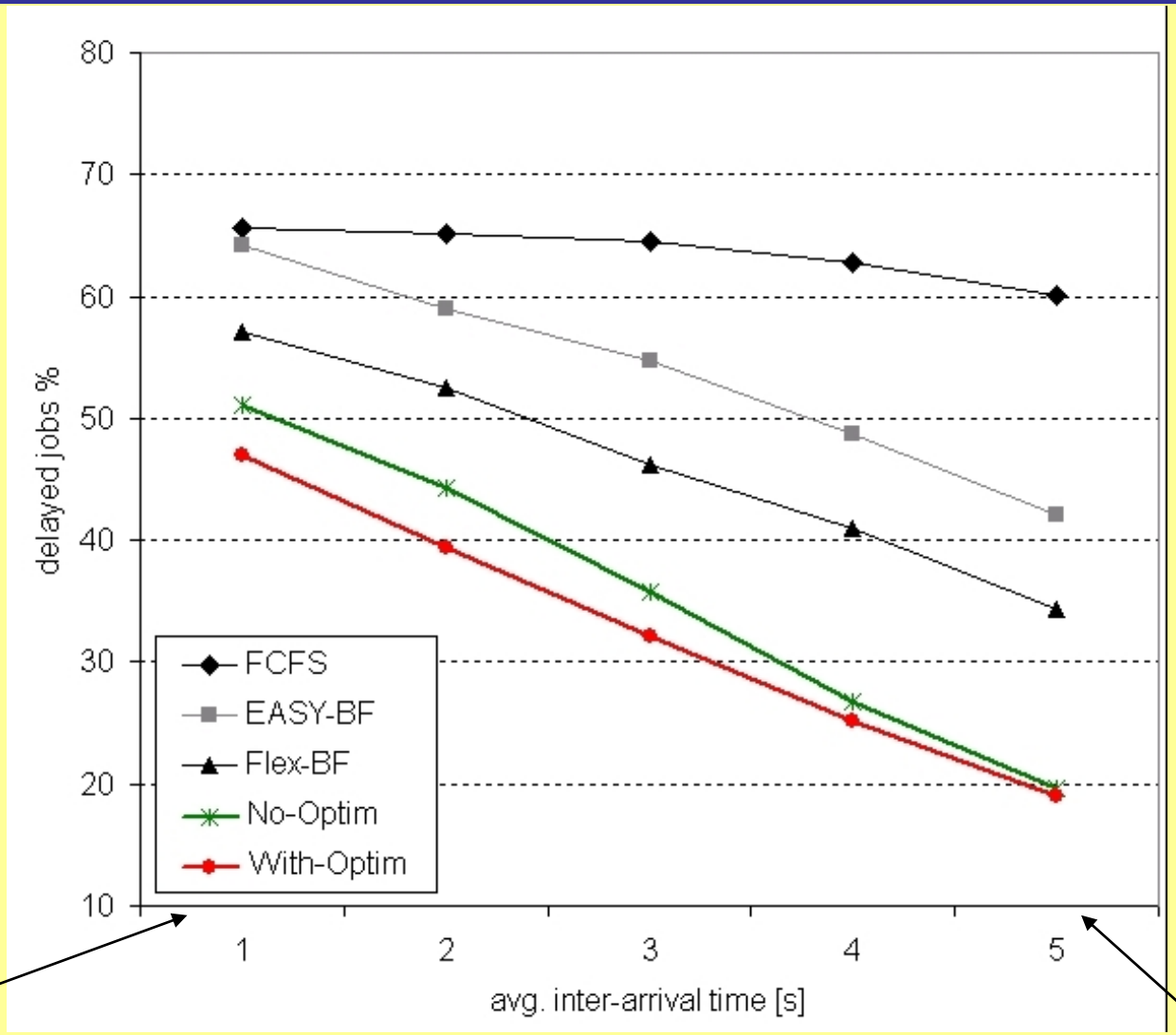
- Schedule optimization
  - To keep/improve the QoS and other goals
  - The use of Local-search based optimization
  - Follows the incremental approach *(local changes)*
  - It is quick and can be stopped at any time (iterative procedure)

Typically used when:
Job completes earlier/later
New machine appears
Better QoS is required
Better utilization is required

Preliminary Results

- Comparison of the proposed techniques
  - Against popular queue-based techniques *(e.g., Backfilling)*
- Using the schedule and event-based approach
  - With and without periodical optimization
  - With dynamic job arrivals
  - With changing QoS requirements
  - Using the incremental approach
    - No recomputing from scratch upon dynamic changes
- Goals
  - QoS *(jobs' deadlines – express the user's expectations)*
  - Machine usage *(expresses the resource owner's expectations)*
  - Reasonable runtime

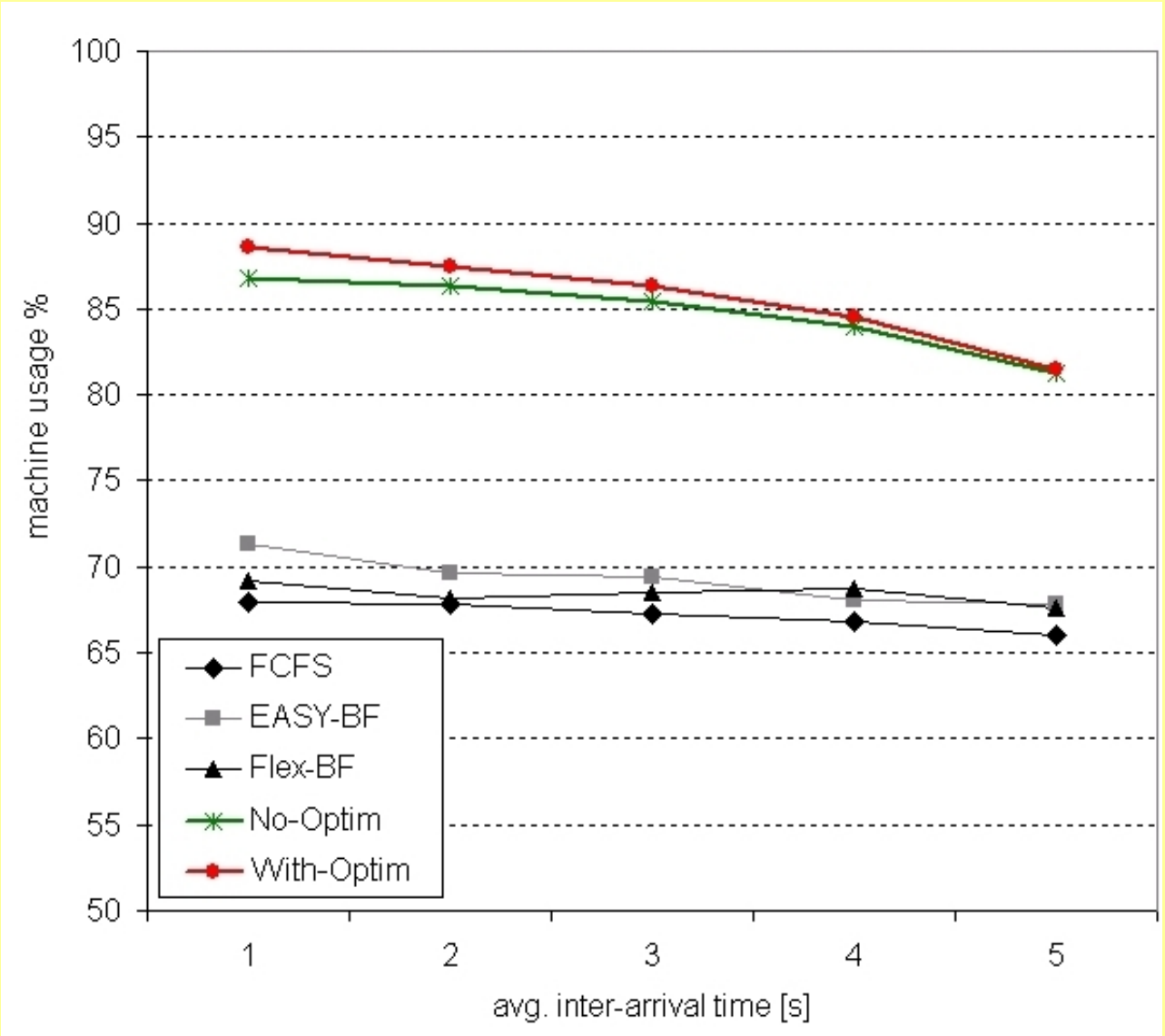
Deadlines



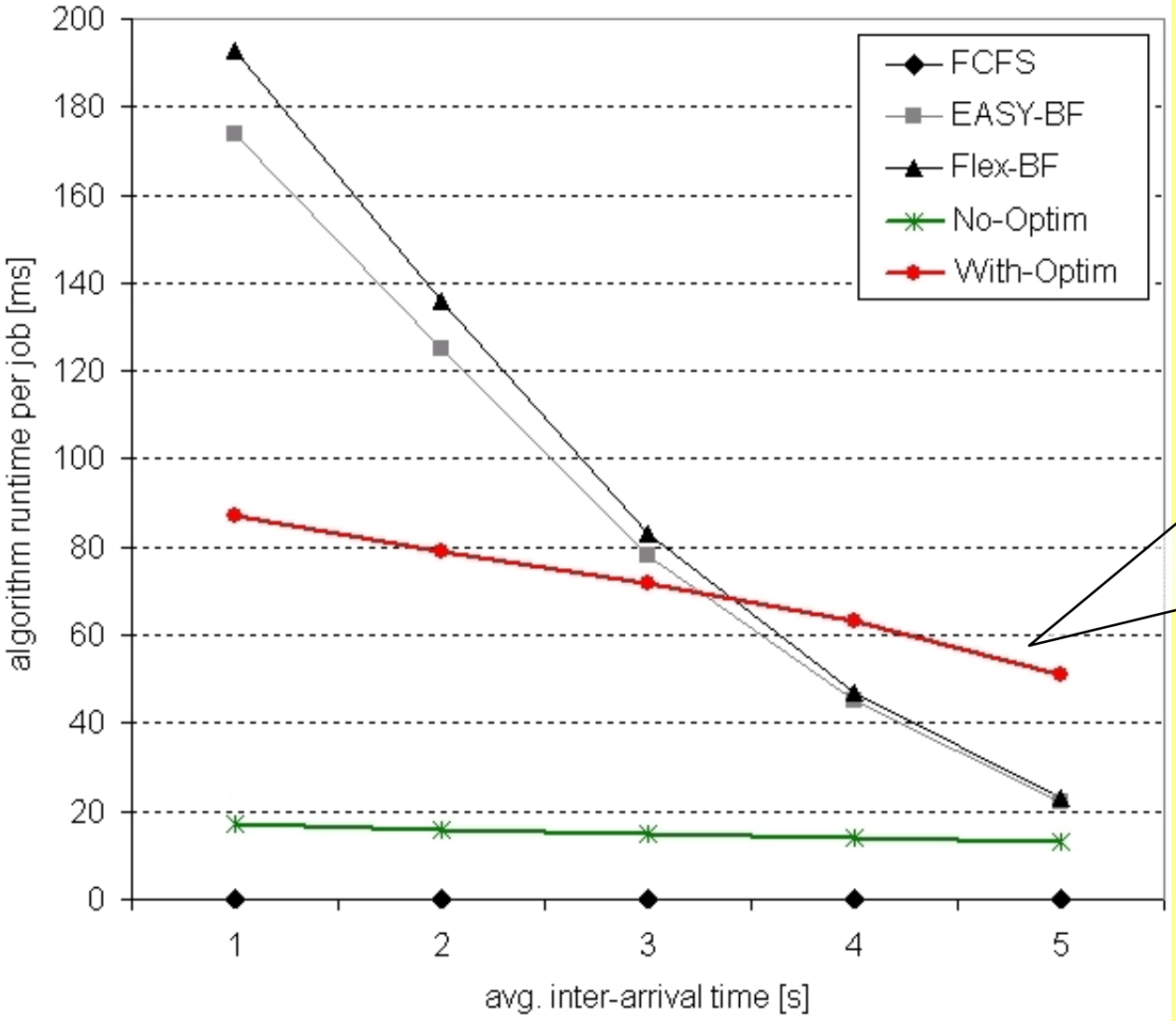
High system load → jobs arrive frequently

Low system load

Machine Usage



Algorithm Runtime



The runtime of the optimization depends on the # of iterations but you can see that it scales more reasonably than both backfilling algorithms

Conclusion & Future Work

- **Conclusion**
  - **Promising preliminary results** for the schedule-based approach
  - **More straightforward optimization** vs. queue-based techniques
  - Several techniques proposed to deal with **uncertainty**
  - Uncertainty is a problem for **almost every technique** that aims to guarantee something

- **Future work**
  - The use of proposed event-based solution using more complex model
    - The use of real data sets using job runtime estimates
    - Dynamic resource changes *(machine restart/failure)*, etc.

Thank You !

Tabu Search

- Local search based algorithm
  - Local changes of existing schedule
  - Moving the jobs within the schedule
- Optimizes the initial schedule
  - By filling “early gaps” with “later jobs”
  - Why?
    - Early gaps would soon cause low machine usage
    - Later jobs are more likely to be delayed
    - Moving later jobs “to the front” may speed up other jobs and prevent them from being delayed
    - Such algorithm can be stopped at “any time” (after any iteration) if fast decisions is required
  - Decisions taken according to the value of **weight** function
    - Weight is computed using the # of delayed jobs and makespan before/after the move
    - weight > 0.0 = accept, weight ≤ 0.0 = reject