Workflows and Scheduling

Frank Röder

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

14-12-2015

Content

Application

- about the applications
  - solving grand challenges
  - modeling, simulations and analysis
  - a lot of data and computing capacity to handle
- how to improve the performance
  - work harder
  - work smarter
  - get help

Application NASA

- with the help of big data and HPC

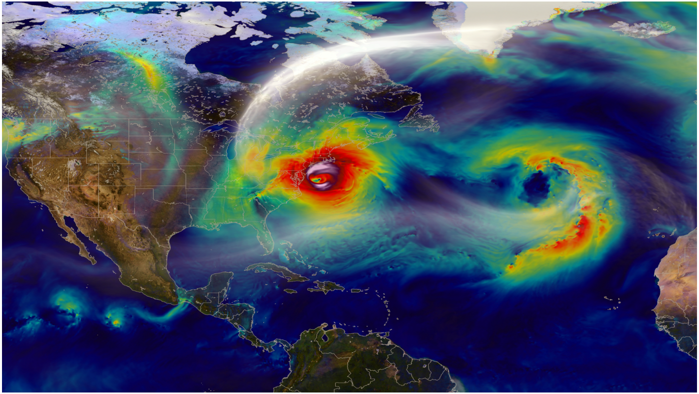


Figure: NASA Climate Sandy Windstorm [1]

What are workflows?

- workflows are
  - the flow and order of work
  - chain of requirements/conditions

  - can be represented as a DAG (directed acycle graph)
  - possible partitioning in to subworkflows

  - explain computational tasks very well
  - helps us to manage the data flow

- the graph for most workflows
  - there are no directed cycles
  - there is no way that a loop from vertex v to vertex u exists
  - its a directed graph with only one direction between two nodes

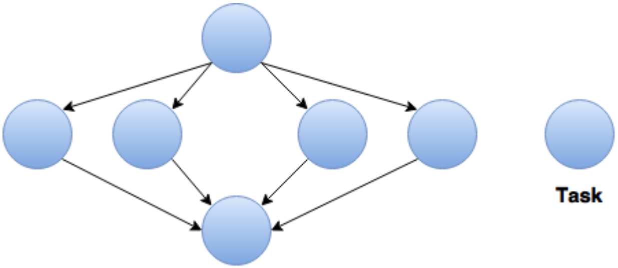


Figure: [6]

- DAG model in scientific workflows
  - each node represents a computational activity
  - the directed edges shows us a dependencies
  - a task is a process, that the user likes to execute
  - a job is a single unit of execution with one or more tasks
  - grouping of similar tasks to jobs

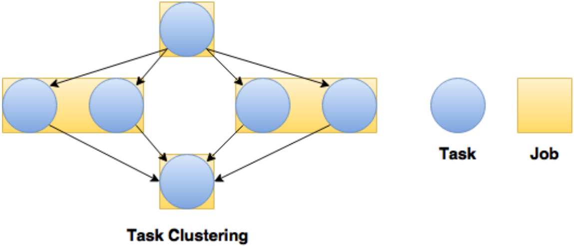


Figure: [6]

- failures in the workflow
  - failures will have impact on the performance
  - task failure as interruption of one task inside a job
  - job failure as interruption of all tasks inside a job
  - need of failure monitoring

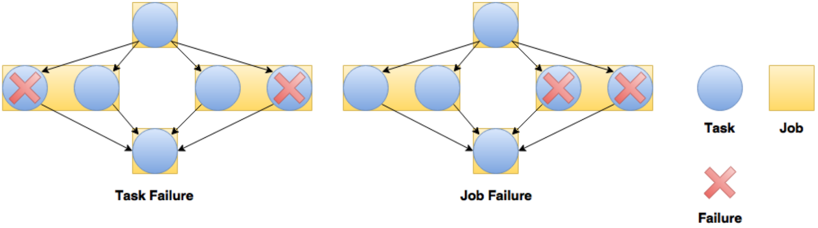


Figure: [6]

- procedure of a workflow
    - prepare the source code, scripts and configuration files
    - collect the input data and make it available on the cluster
    - maybe parallel input/reading of the data (bottleneck)

    - run one or even more independent sets of experiments
    - for example with MPI where the parts refer on each other
    - collect the calculated important data and maybe store it

    - visualize/ analyze the data
    - archive the result
    - parallel writing of results on the storage

example of an workflow

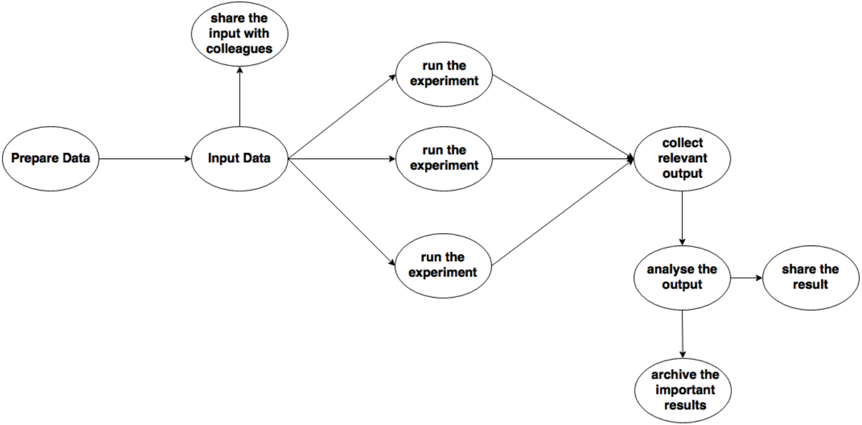


Figure: workflow as a DAG [5]

Scheduling in general

- scheduling
  - plan of time
  - controls the when of executions
  - plan of access to resources (allocation...)
  - core can be about performance
- scheduling-algorithms
  - make decisions about resources and granting time
  - rescheduling during runtime
  - about performance to keep the workflow going
  - have to be highly available (own computing unit)
  - maybe predictions about expected runtimes

- **static**
  - predict the runtime and execute without possible interruption
  - strict list of execution and schedule never changes, even if some unit has nothing to do
- **dynamic**
  - possible reschedule while running
  - realtime information about the environment
  - make decision with those actual information
  - important step of look-ahead

Mapping of an workflow - scheduling of execution

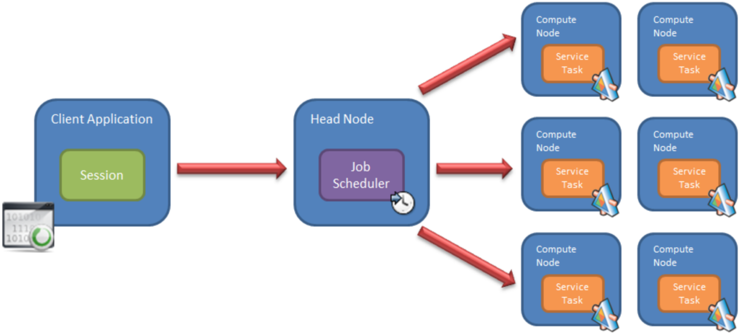


Figure: scheduling [microsoft.com]

HEFT-Algorithm Part 1

- HEFT as scheduling algorithm
  - scheduling optimized for performance
  - provides us with up-to-date information

```
T - set of all tasks in the workflow
E - set of all dependencies
R - set of all available resources
(t1,t2) - dependence between task t1 and t2
time(t,r) - execution time of task t on resource r
time(e,r1,r2) - data transfer time of data between r1 and r2
```

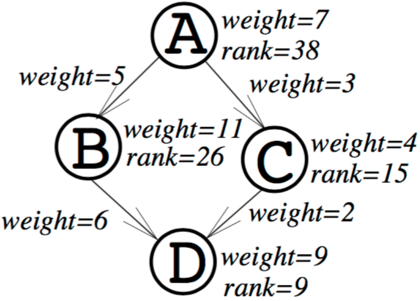
Listing 1: pseudocode 4

HEFT-Algorithm Part 2

```
//Weight phase
for each t ∈ T do
    w(t) = (∑_{r∈R} time(e,r1,r2)) / R
for each e ∈ E do
    w(e) = (∑_{r1,r2∈R,r1≠r2} time(e,r1,r2)) / (R(R−1))
//Ranking phase
take the max of sum (w(t),w(e)) from bottom
    ↪ to the top
ranking = sort(T,rank)
//Mapping phase
for i ranking downto 1 do
    t = ranking[i]
Find resource r ∈ R - min(finish_time(t,r))
Schedule t to r
Mark r as reserved until finish_time(t,r)
```

Listing 2: pseudocode 4

HEFT in action



	R1	R2	R3	*avg*
A	5	8	8	7
B	9	13	11	11
C	3	4	5	4
D	7	10	10	9

execution times on different resources

	R1–>R2	R1–>R3	R2–>R3	*avg*
A–>B	6	4	5	5
A–>C	4	2	3	3
B–>D	7	4	7	6
C–>D	1	1	4	2

data transfer time between different resources

Figure: [4]

Conclusion Heft

- Why to use the HEFT-Algorithm?
  - best algorithm for scheduling in most cases
  - uses the order of executions as fact
  - can handle complicated DAGs

Myopic-algorithm

```
T - set of all tasks in the workflow
NT = T
while NT ≠ ∅ do

    Find task t ∈ NT with
        ↪ min(earliest_starting_time(t))

    Find resource r ∈ R : min(finish_time(t,r))

    Schedule t to r

    Mark r as reserved until finish_time(t,r)

    NT = NT \ {t}
end
```

Listing 3: pseudocode 4

Conclusion Myopic

- Why to use the Myopic-algorithm?
  - inexpensive algorithm based on local optimal decisions
  - can produce quite accurate results for simple graphs

  - won't provide us with full-graph analyses
  - won't use any order of tasks

... even workflows can be simulated

- we can import our workflow as DAG file
- it will list up our tasks
- will use a Failure Generator and a Failure Monitor
- overhead modeling

- get a idea about our workflow before
- got to know which scheduling fits the most

- workflowsim.org

Different approach

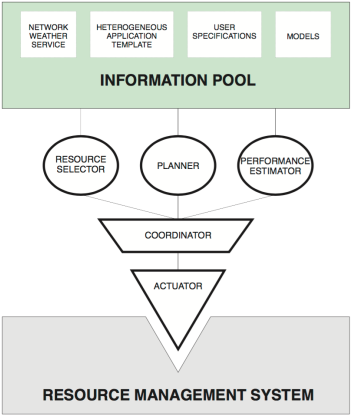


Figure: AppLeS

Summary

- Resource Hungry Applications
  - big data, a lot of computational work
  - thinking about efficiency and performance increase
- Workflows
  - order of work as DAG
  - grouping of similar task into a job
- Scheduling
  - mapping of workflow steps to resources
  - need of intelligent algorithms to find good solutions

- Sources
  - Link: http://www.nas.nasa.gov/SC13/assets/images [1]
  - Link: http://pegasus.isi.edu [2]
  - Link: http://www.workflowsim.org [3]
  - Link: http://www.sigmod.org/publications/sigmod-record [4]
  - Link: https://wikis.nyu.edu/display [5]
  - Link: http://de.slideshare.net/WeiweiChen/workflowsim-escience12-14674703 [6]
  - Link: http://citeseerx.ist.psu.edu [7]
  - Link: http://perso.univ-lyon1.fr [8]