

Faithful Performance Prediction of a Dynamic Task-based Runtime System, an Opportunity for Task Graph Scheduling

Samuel Thibault, Luka Stanisic, Arnaud Legrand

▶ **To cite this version:**

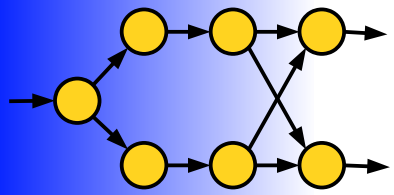
Samuel Thibault, Luka Stanisic, Arnaud Legrand. Faithful Performance Prediction of a Dynamic Task-based Runtime System, an Opportunity for Task Graph Scheduling. SIAM PP 2020 - SIAM Conference on Parallel Processing for Scientific Computing, Feb 2020, Seattle, United States. hal-02943753

HAL Id: hal-02943753
https://inria.hal.science/hal-02943753

Submitted on 20 Sep 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

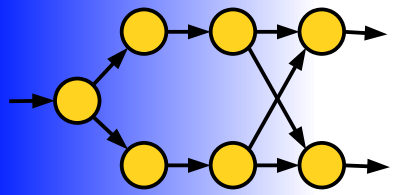
L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Faithful Performance Prediction of a Dynamic Task-based Runtime System, an Opportunity for Task Graph Scheduling

Samuel Thibault, Luka Stanisic, Arnaud Legrand

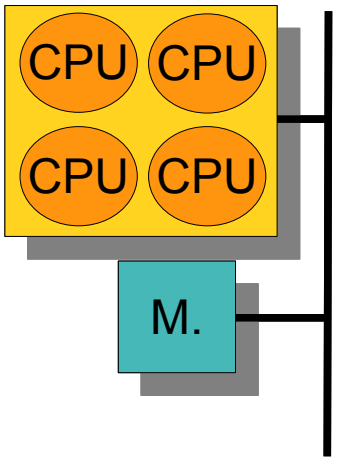
LaBRI université de BORDEAUX Inria UGA Univ. Grenoble Alpes LIG

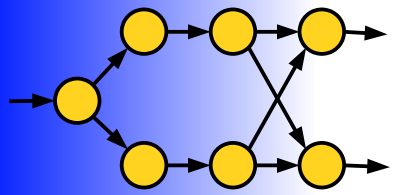


High-Performance Computing

Combination of hardware

- Classical CPUs

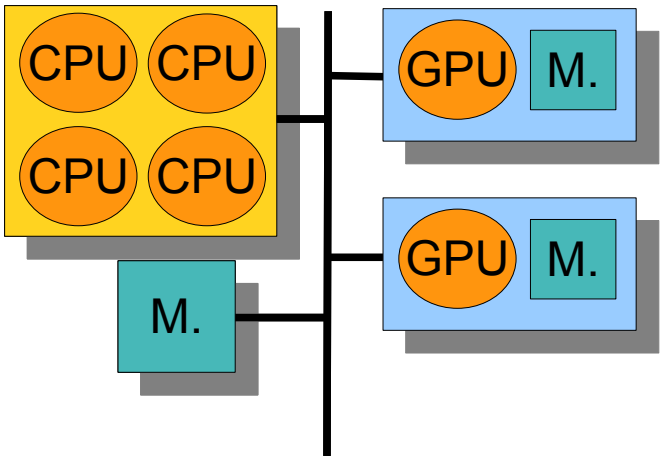


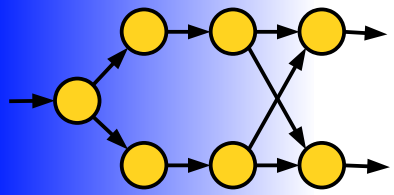


High-Performance Computing

Combination of hardware

- Classical CPUs
- Accelerators: GPUs

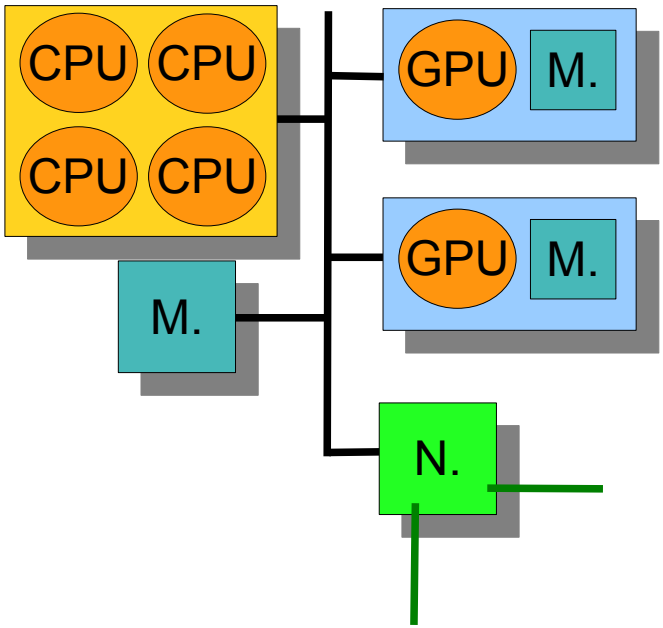


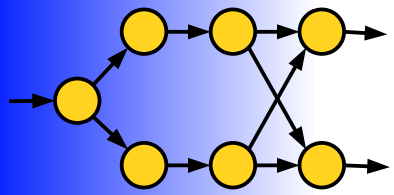


High-Performance Computing

Combination of hardware

- Classical CPUs
- Accelerators: GPUs
- Network card

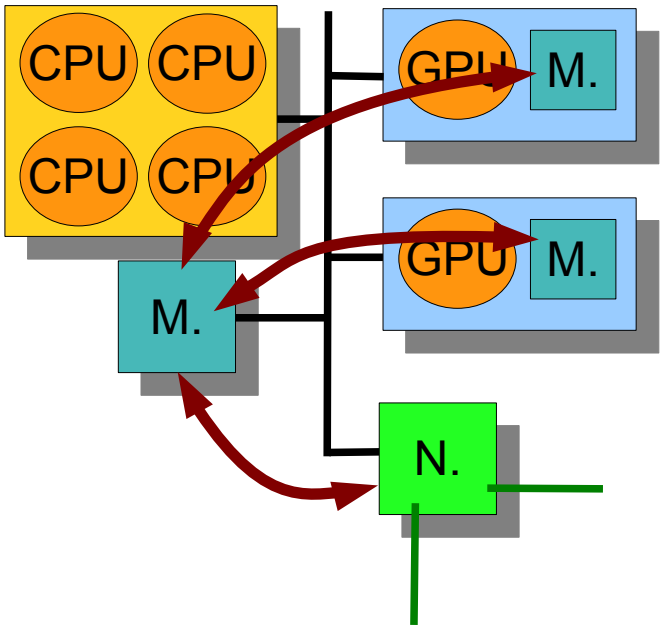


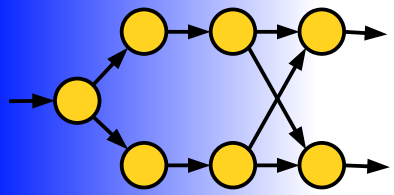


High-Performance Computing

Combination of hardware

- Classical CPUs
- Accelerators: GPUs
- Network card
- Data transfers





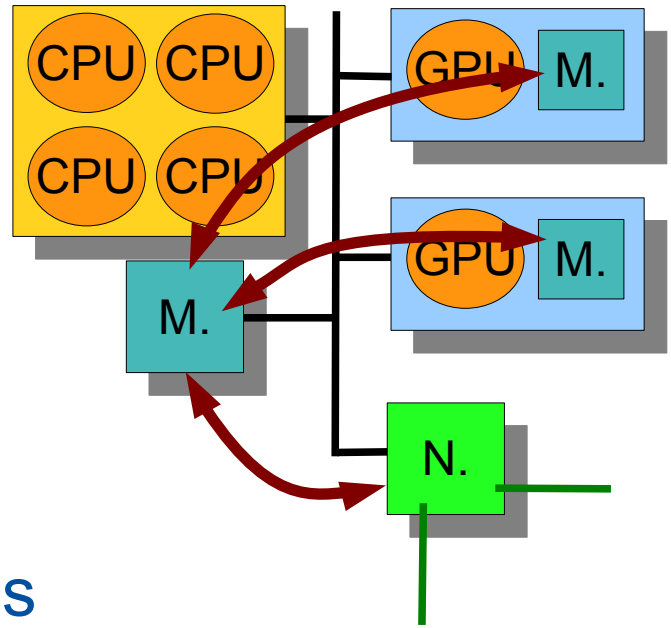
High-Performance Computing

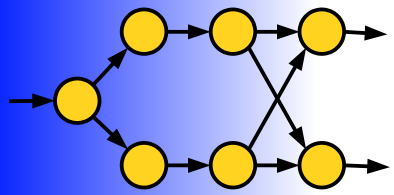
Combination of hardware

- Classical CPUs
- Accelerators: GPUs
- Network card
- Data transfers

Managing it all by hand?!

- New machine with more GPUs

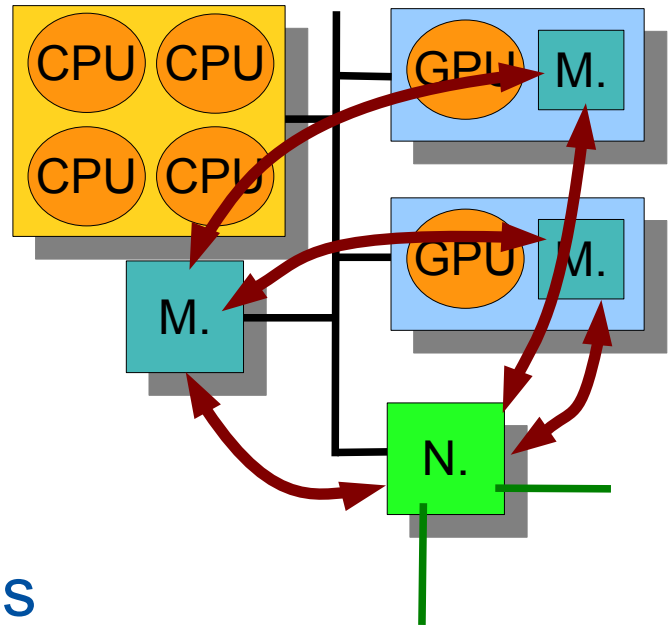




High-Performance Computing

Combination of hardware

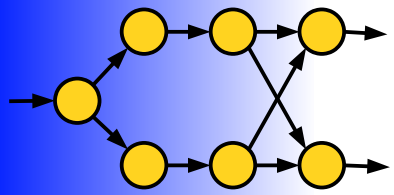
- Classical CPUs
- Accelerators: GPUs
- Network card
- Data transfers



Managing it all by hand?!

- New machine with more GPUs
- New transfer possibilities
- …

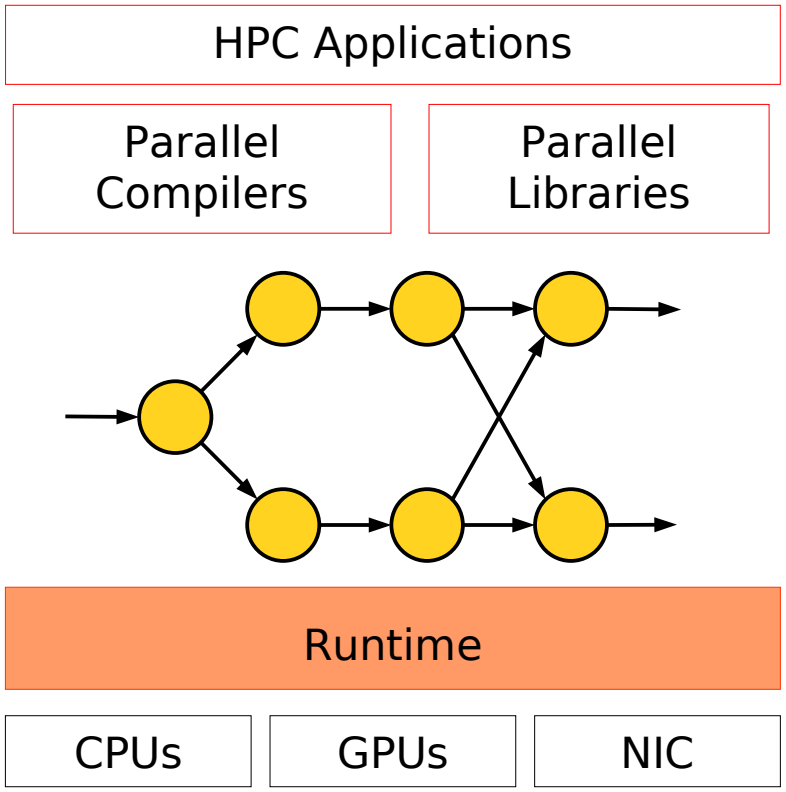
→ Runtime systems

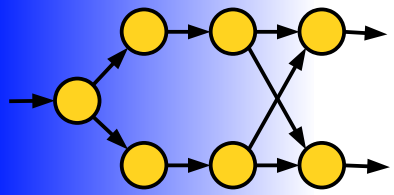


High-Performance Computing

A recent actual trend towards runtime systems & task graphs

- Applications
  - Submit a task graph
- Runtime system
  - Manages & optimizes execution
  - Copes with memory limitation
  - …

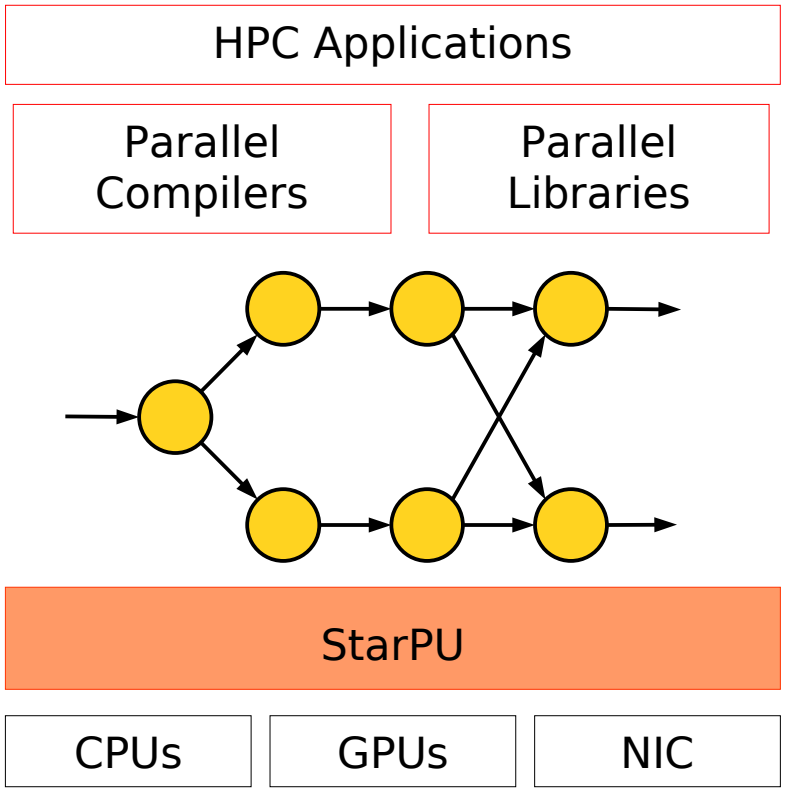


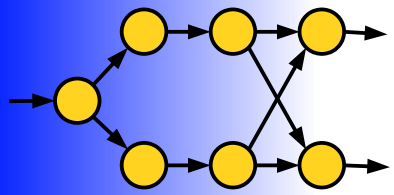


High-Performance Computing

The StarPU runtime system

- Started in 2008
  - PhD Thesis of Cédric Augonnet
- StarPU main core ≈ 70k lines
- Written in C
- Open Source
  - Release under LGPL
  - Sources freely available
    - git repository and nightly tarballs
    - See https://starpu.gforge.inria.fr/
  - Open to external contributors
- [Europar'09,CCPE'11]: > 1200 citations

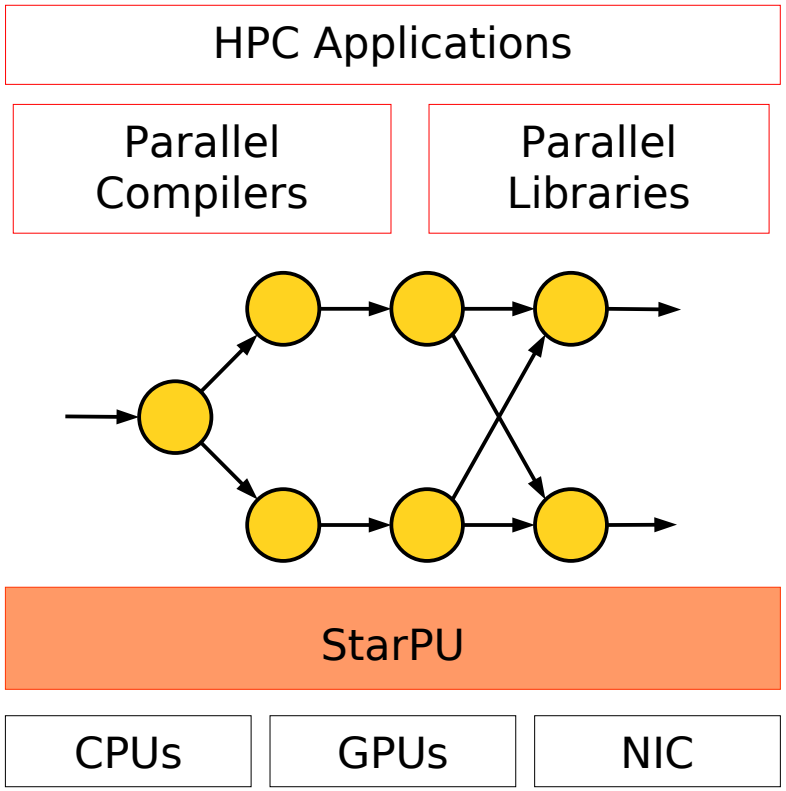




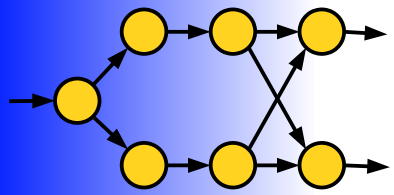
High-Performance Computing

Difficulties shifted

- Applications
  - Decide task graph structure
  - Decide task granularity
- Runtime system
  - Scheduling heuristics
  - Eviction heuristics
  - …



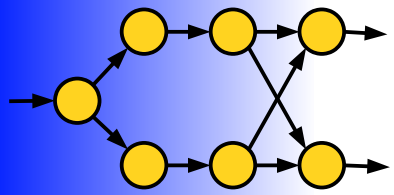
And debug…



High-Performance Computing

Confirming performance improvements with measurements

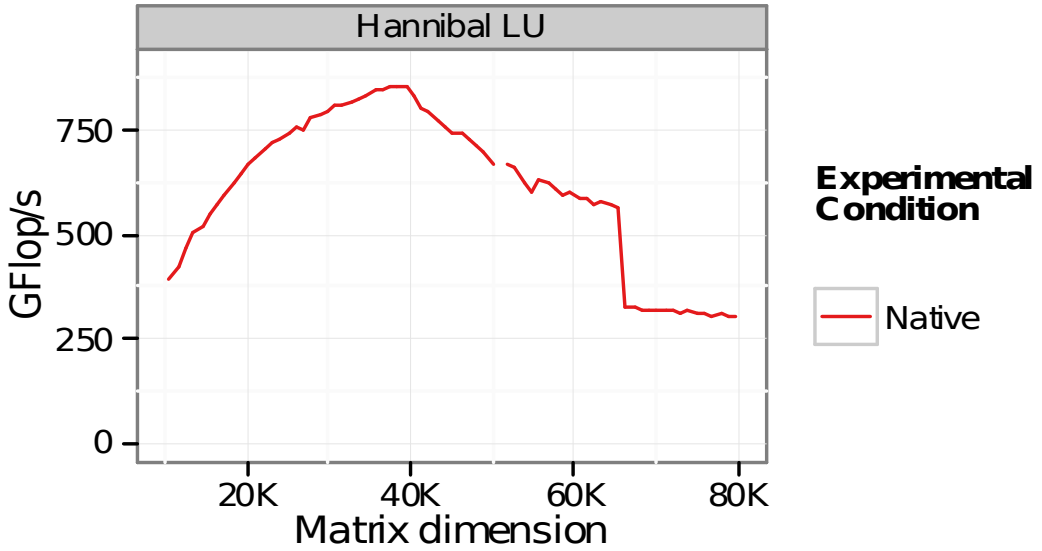
- Robust? Platform stability
  - Rogue processes
  - Faulty cables
  - Software upgrades
  - Firmware upgrades
- Repeatable? Platform availability
  - CPU.hour allocation
  - Decommissioned platform
- Reproducible? Different platforms
  - Getting accounts on various systems
  - Install software stack there

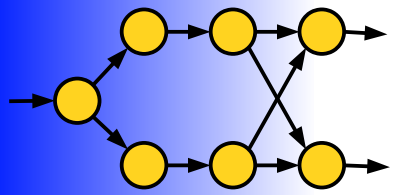


High-Performance Computing

For instance, LU factorization on Hannibal machine

- Odd behavior, stable
- Old GPU, old machine
- Now decommissioned





High-Performance Computing

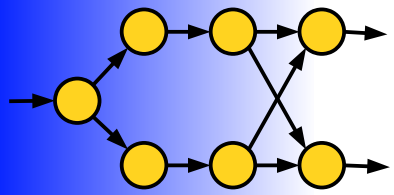
Confirming performance improvements with **simulation**

- Platform stability
  - Deterministic execution

    → Reproducible performance, and bugs!
- Platform availability
  - Can run on commodity laptop

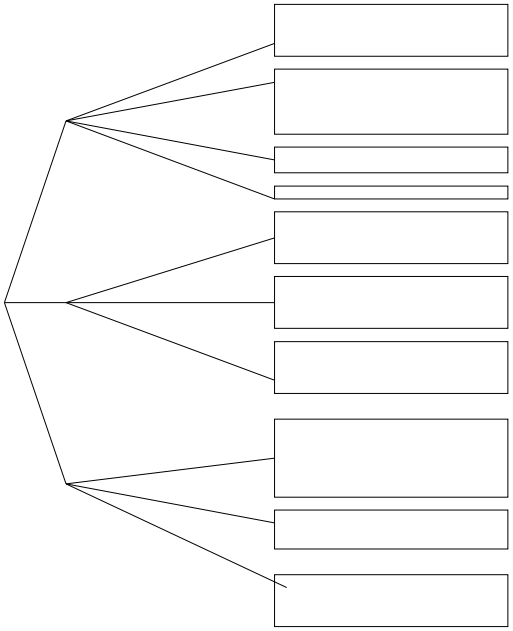
    → No need for platform reservation
- Different platforms
  - Sharing platform modeling

    → Predicting performance



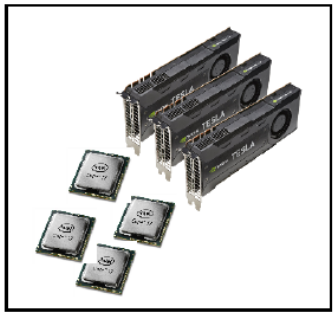
Simulating a runtime system

- Combine emulation and simulation at task level
  - Task graph control code emulated
    - Both runtime and application main code executed unmodified
  - Task computation and data transfers simulated
    - Performance models

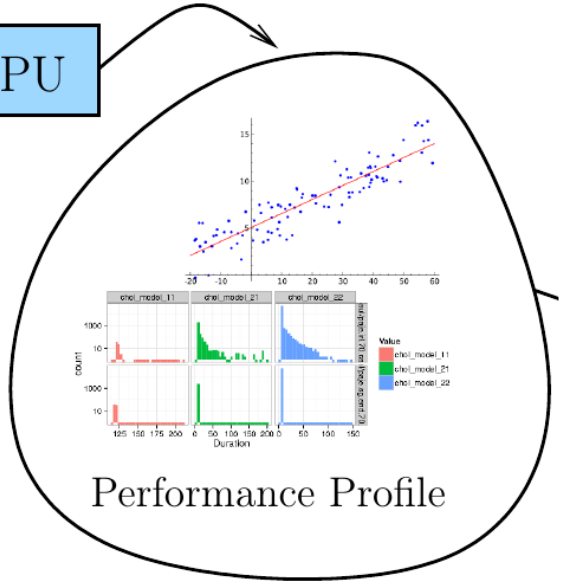


Simulation with SimGrid

Calibration



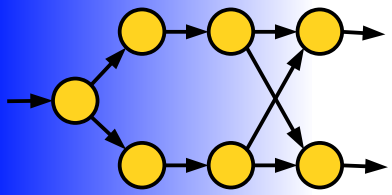
App | StarPU



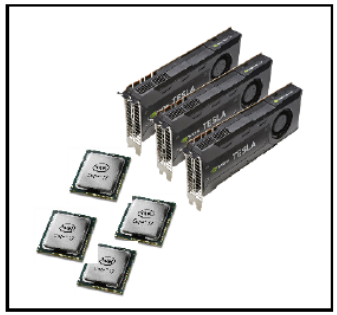
With A. Legrand and L. Stanisic

Run once!

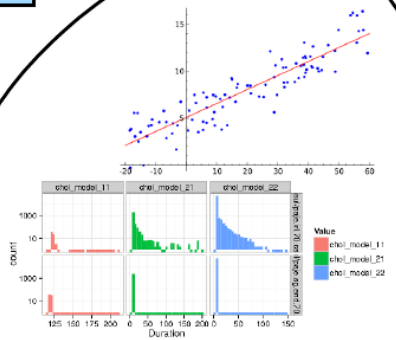
Simulation with SimGrid



Calibration



App | StarPU



Performance Profile

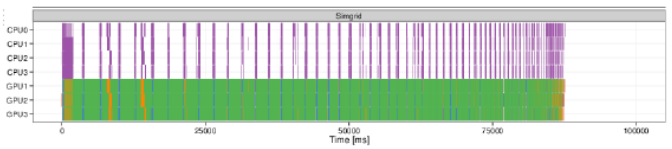
Run once!

Simulation



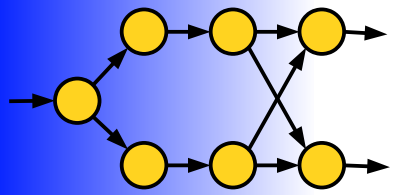
App | StarPU

SimGrid



Quickly Simulate Many Times

With A. Legrand and L. Stanisic



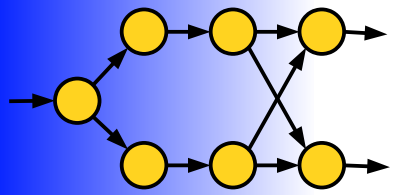
Simulation with SimGrid

The runtime tells SimGrid about

- Threads doing virtual sleeps corresponding to tasks
- Data transfers over the PCI bus

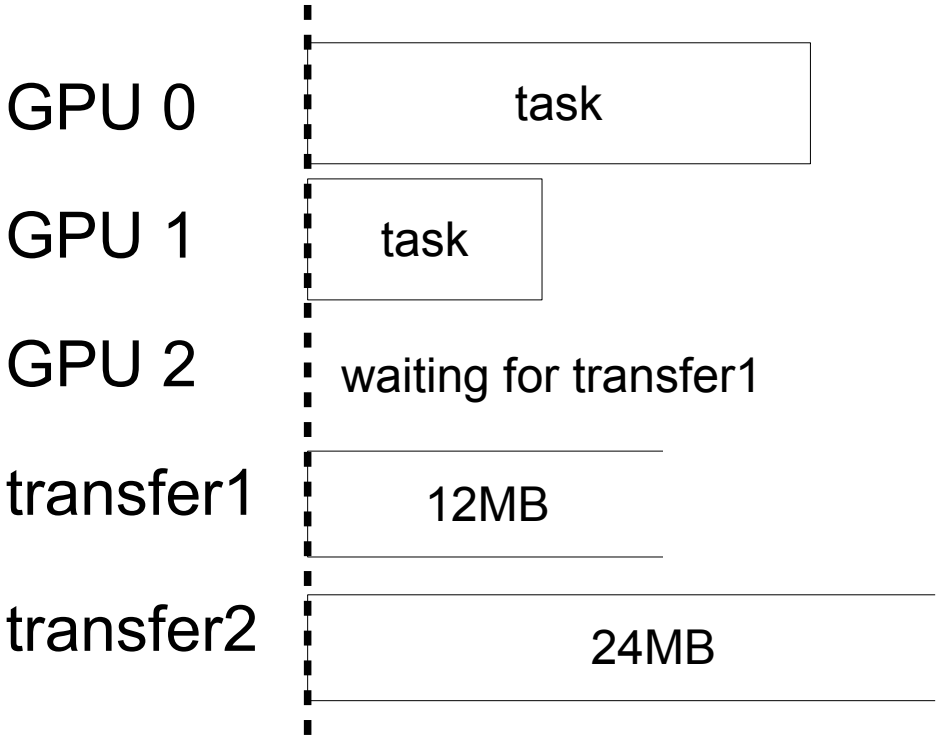
SimGrid manages

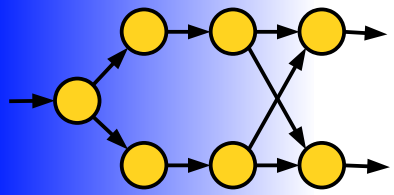
- Synchronization between threads, and with data transfers
- Simulated PCI bandwidth shared between data transfers



Simulation with SimGrid

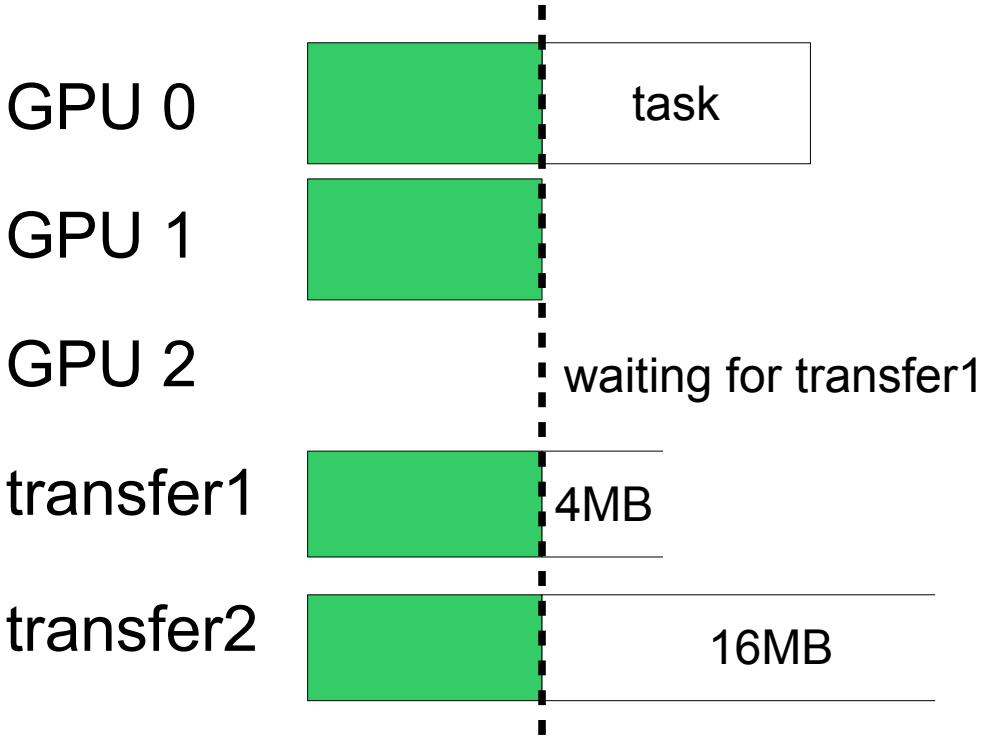
- At each timestep, SimGrid determines next event to trigger
  - thread sleep completion,
  - or data transfer completion
- And advances simulated time accordingly

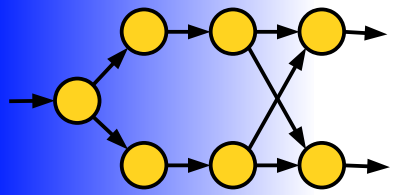




Simulation with SimGrid

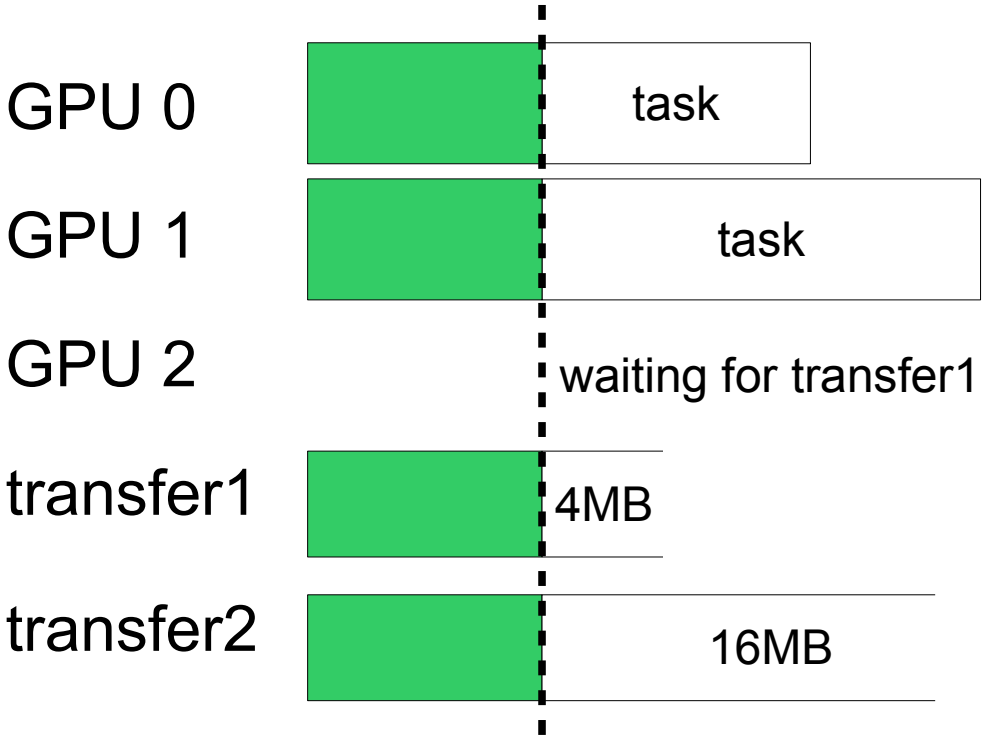
- At each timestep, SimGrid determines next event to trigger
  - thread sleep completion,
  - or data transfer completion
- And advances simulated time accordingly

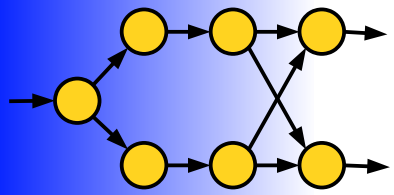




Simulation with SimGrid

- At each timestep, SimGrid determines next event to trigger
  - thread sleep completion,
  - or data transfer completion
- And advances simulated time accordingly

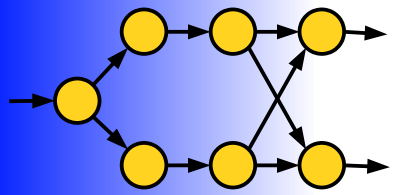




Simulation with SimGrid

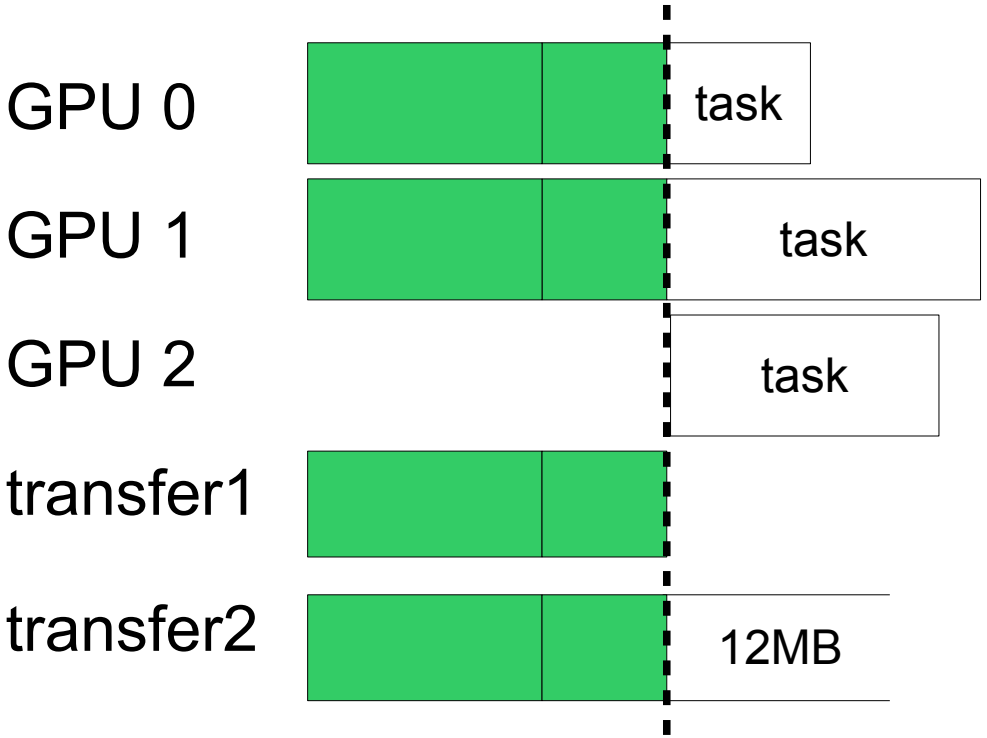
- At each timestep, SimGrid determines next event to trigger
  - thread sleep completion,
  - or data transfer completion
- And advances simulated time accordingly

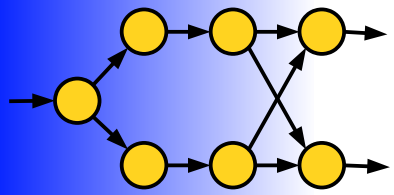




Simulation with SimGrid

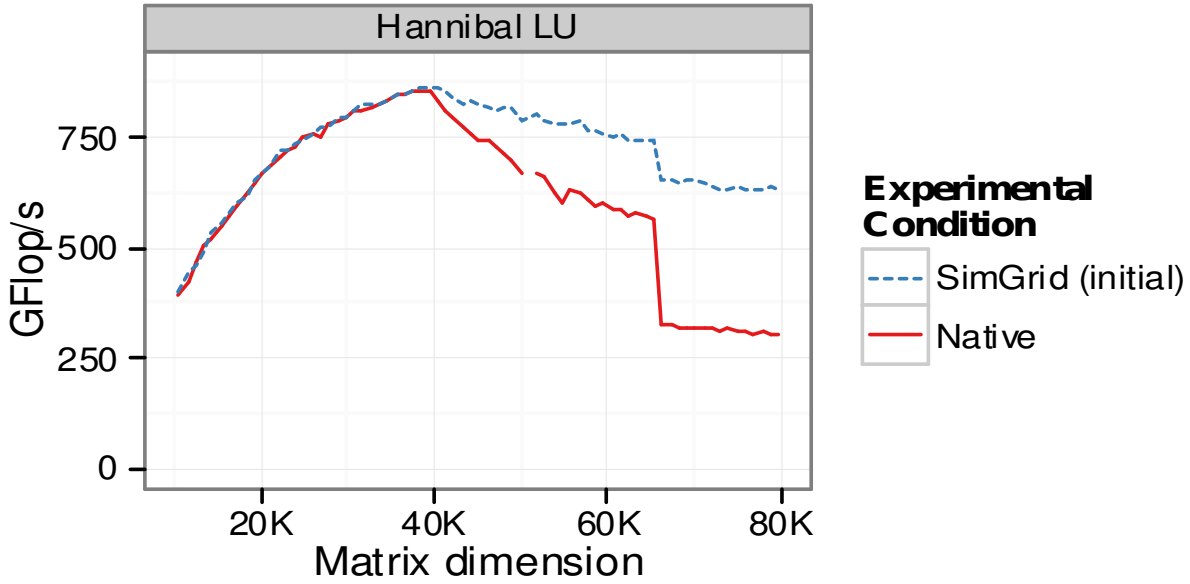
- At each timestep, SimGrid determines next event to trigger
  - thread sleep completion,
  - or data transfer completion
- And advances simulated time accordingly





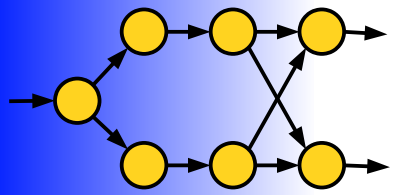
Simulation with SimGrid

Result on LU factorization on Hannibal machine



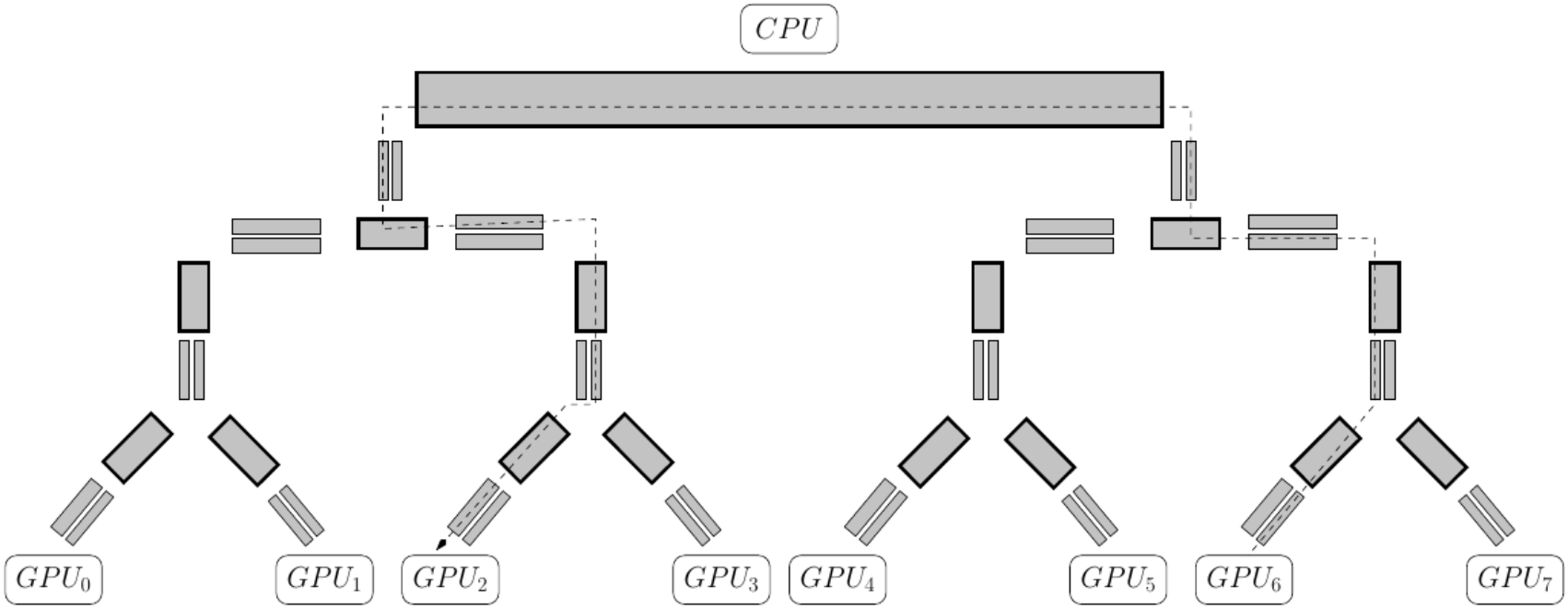
Accurate until matrix too big for all data transfers to overlap

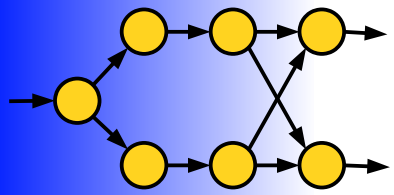
Take GPU memory limitation into account (2.5 GB)



Refining data transfers

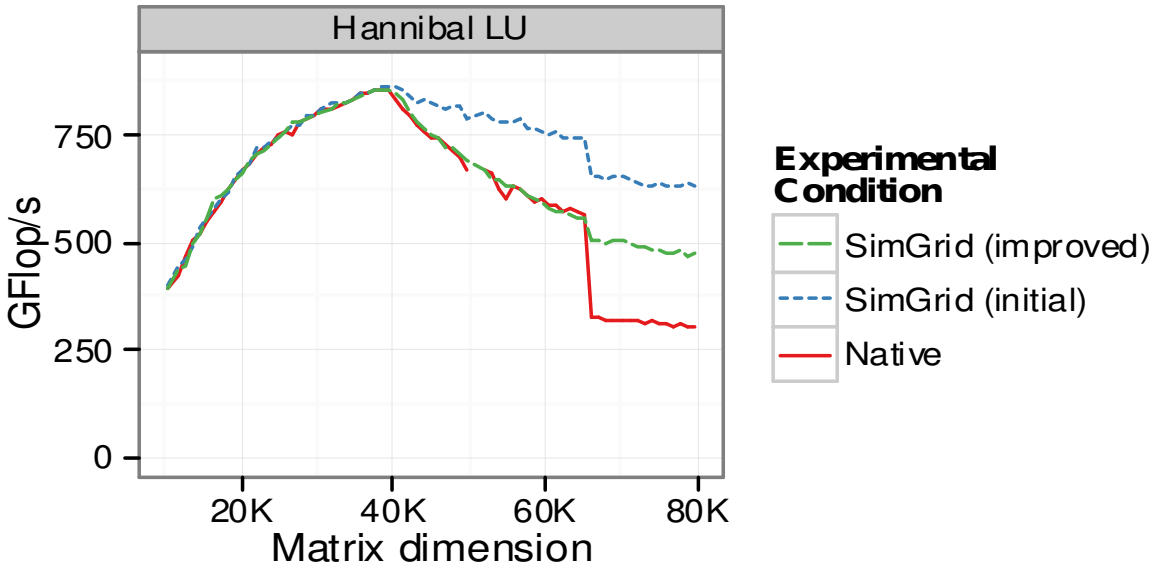
- Improve PCI network model
  - Support direct GPU-GPU transfers
  - Generated by StarPU from measurements on target system
  - And even PCI tree from hwloc information



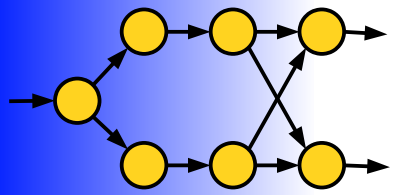


Refining data transfers

Result on LU factorization on Hannibal machine



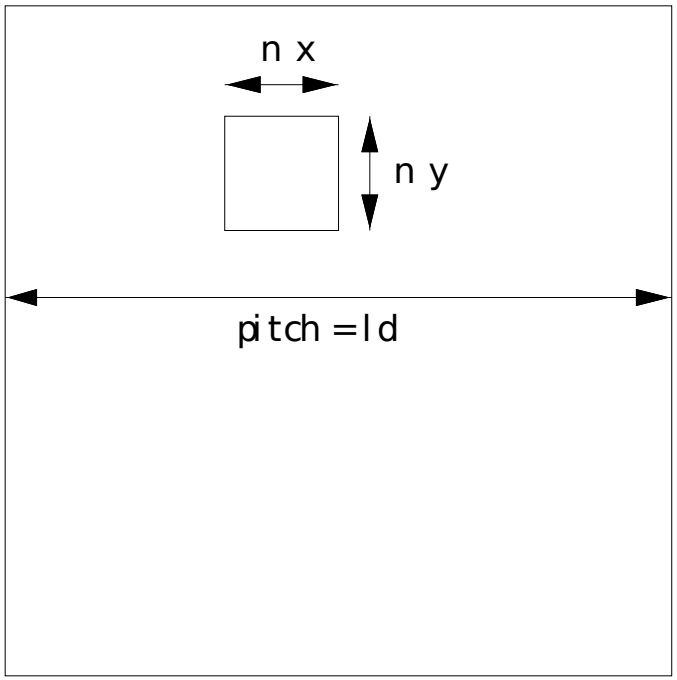
- Way more accurate
- Still odd behavior beyond 66K

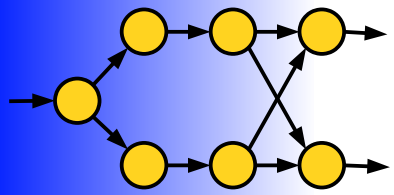


Pitch issue with Quadro FX5800

Tile within big matrix → pitch (LD: leading dimension)

Old QuadroFX5800 does not work well with pitch ≥ 264KB

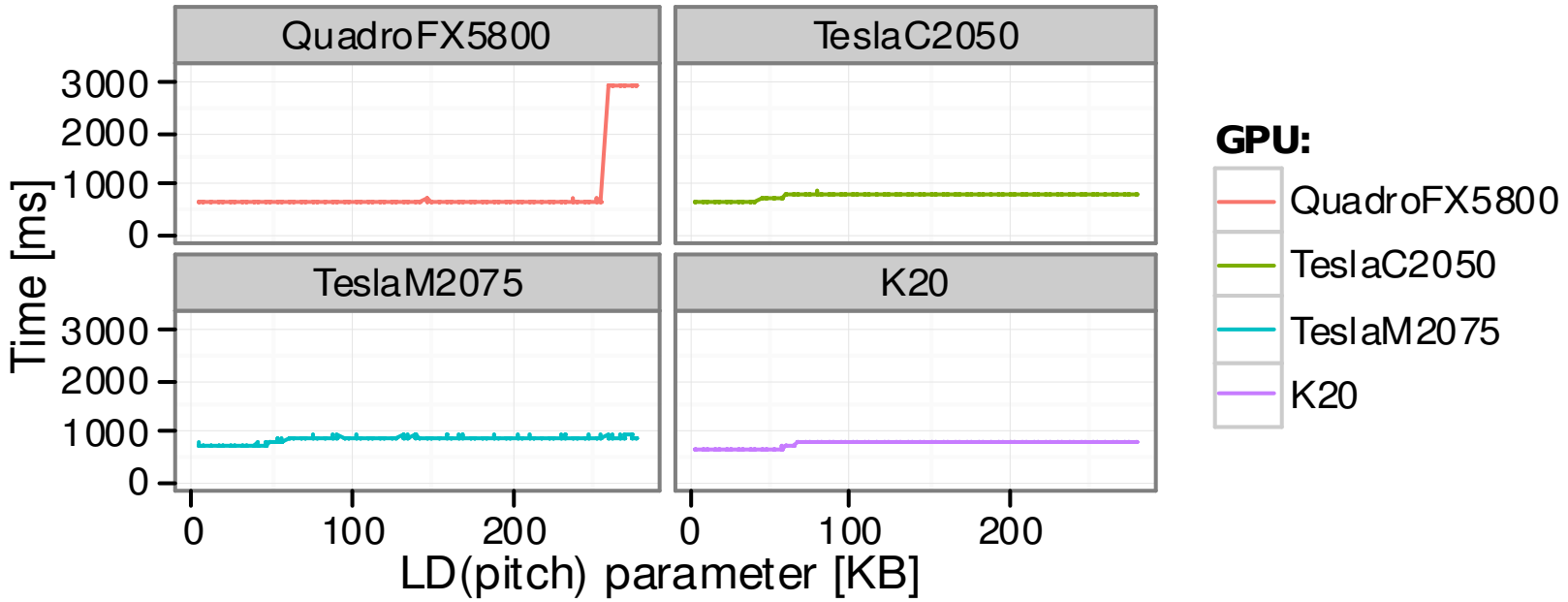


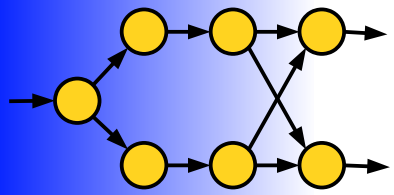


Pitch issue with Quadro FX5800

Tile within big matrix → pitch (LD: leading dimension)

Old QuadroFX5800 does not work well with pitch ≥ 264KB

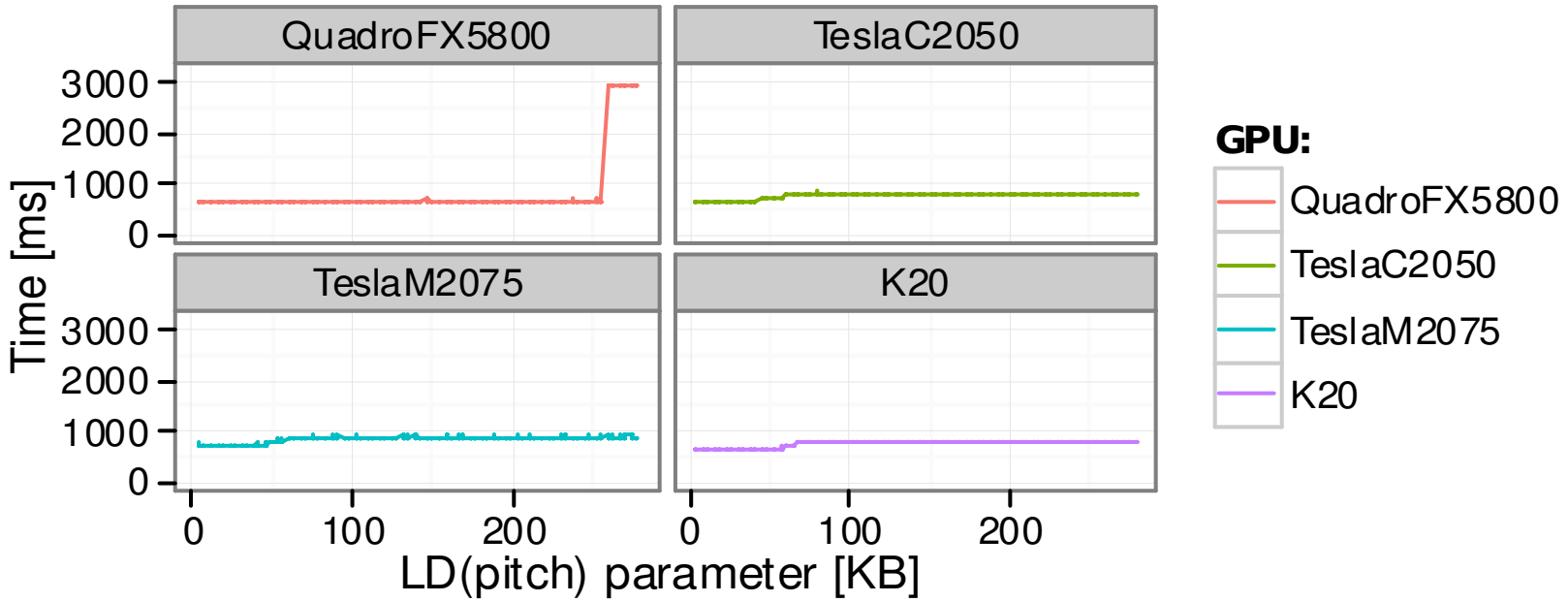




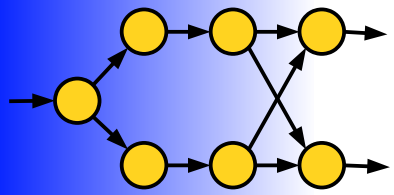
Pitch issue with Quadro FX5800

Tile within big matrix → pitch (LD: leading dimension)

Old QuadroFX5800 does not work well with pitch ≥ 264KB

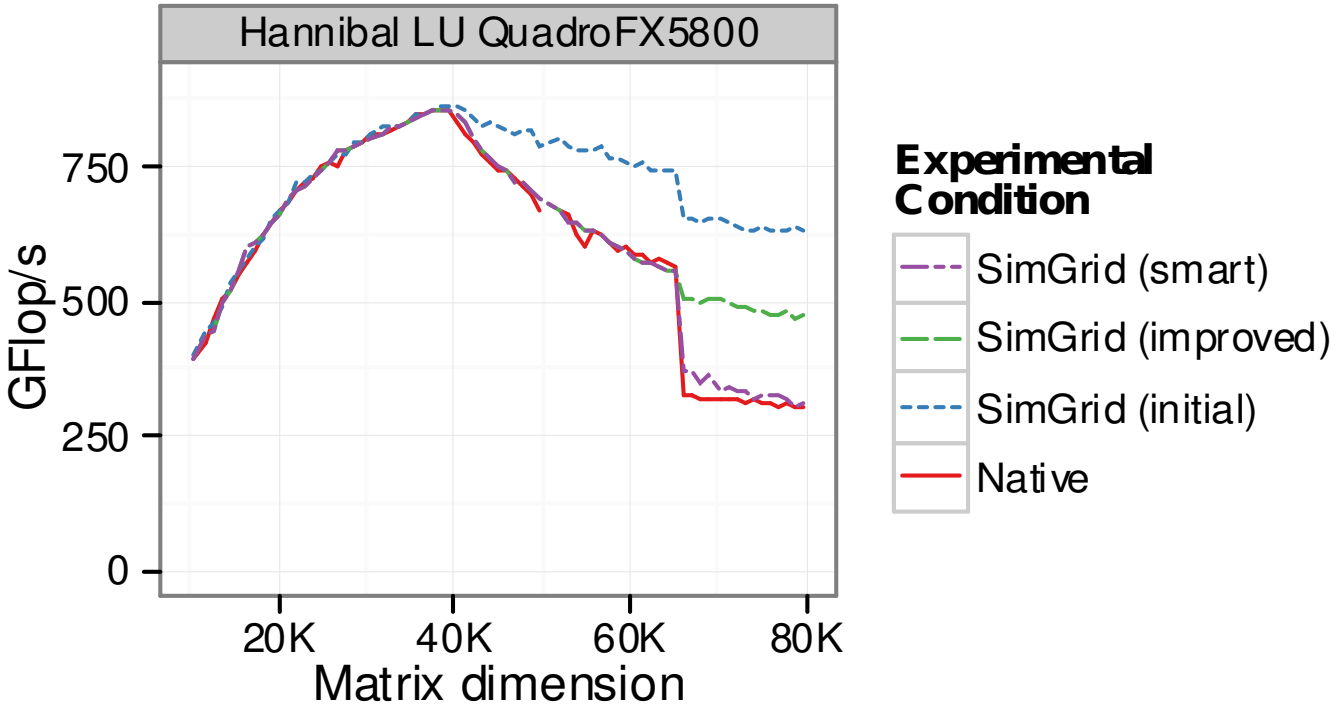


→ Update transfer performance model accordingly

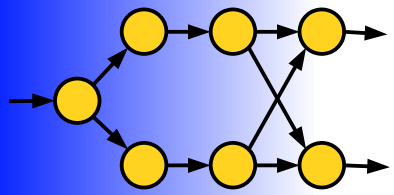


Refining data transfers

Result on LU factorization on Hannibal machine

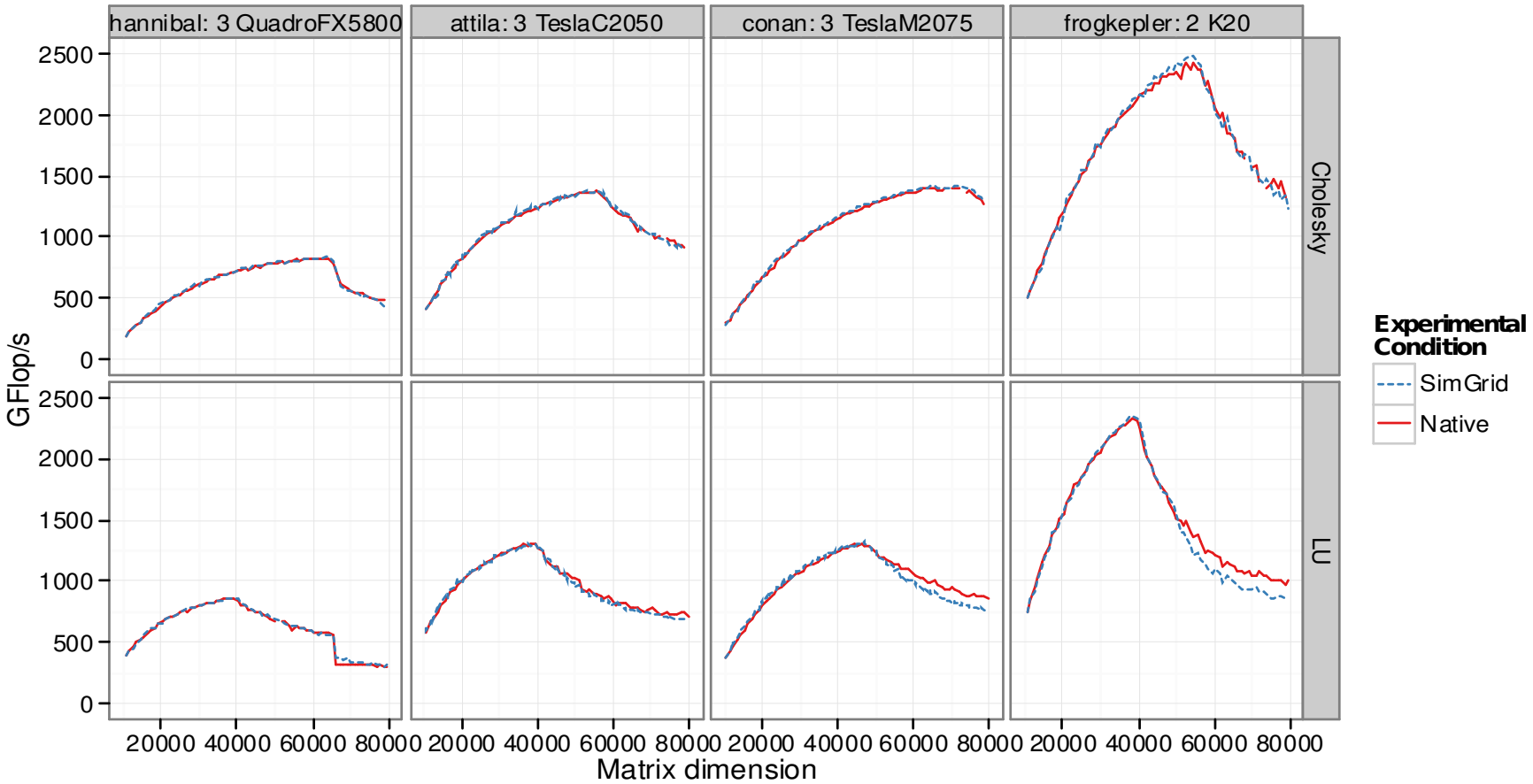


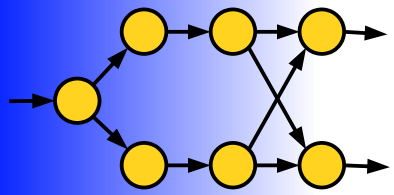
Simulation often reveals real bugs



Results

Various system, various applications

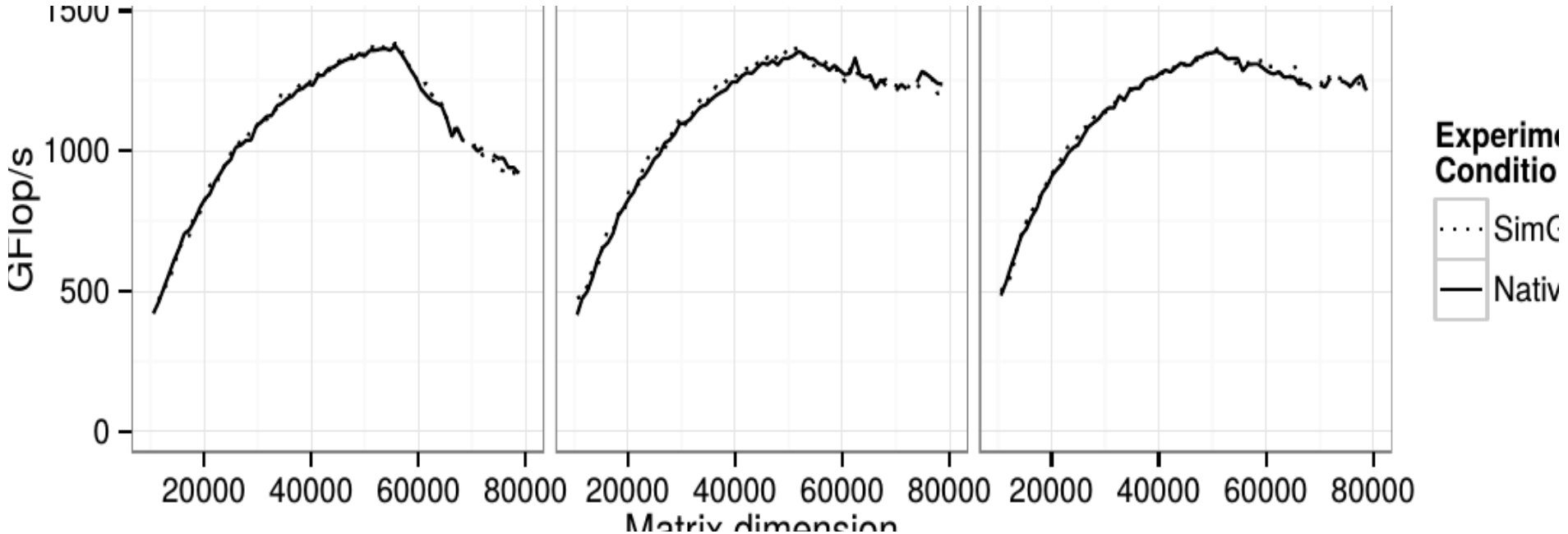


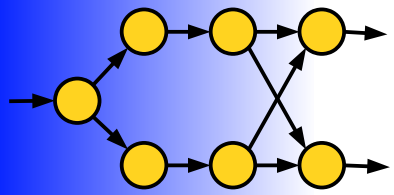


Results

Various scheduling algorithms

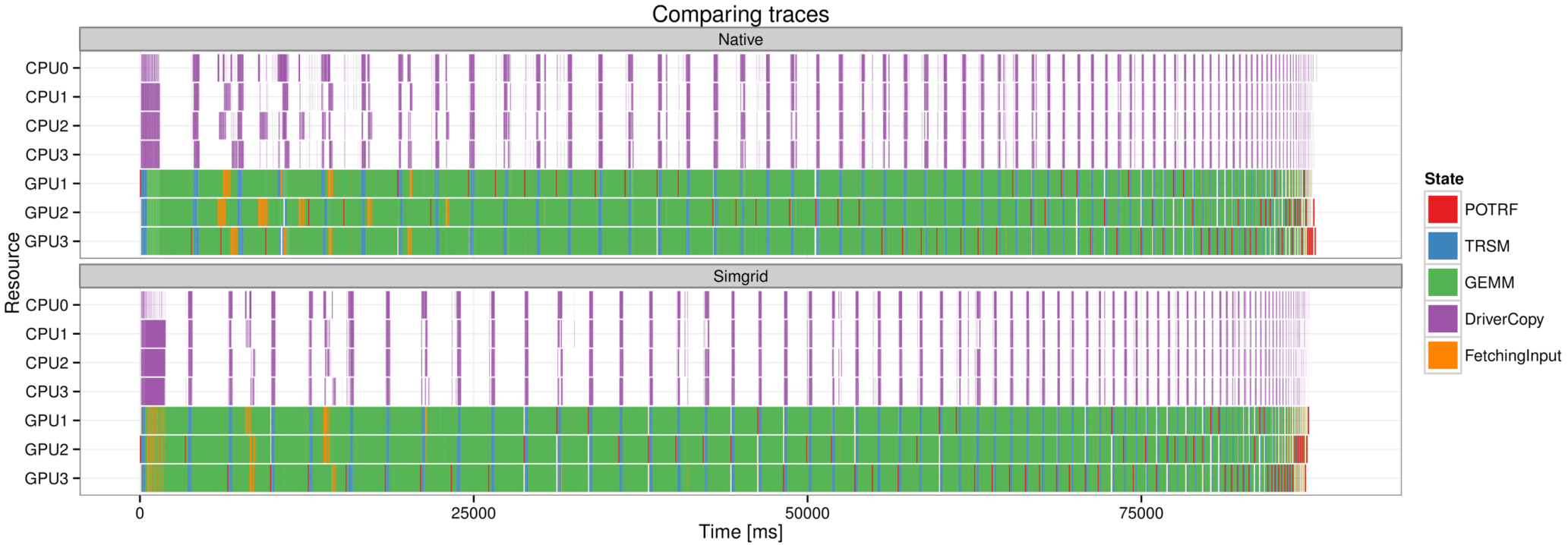
(Cholesky on attila)

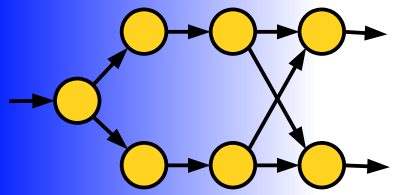




Results

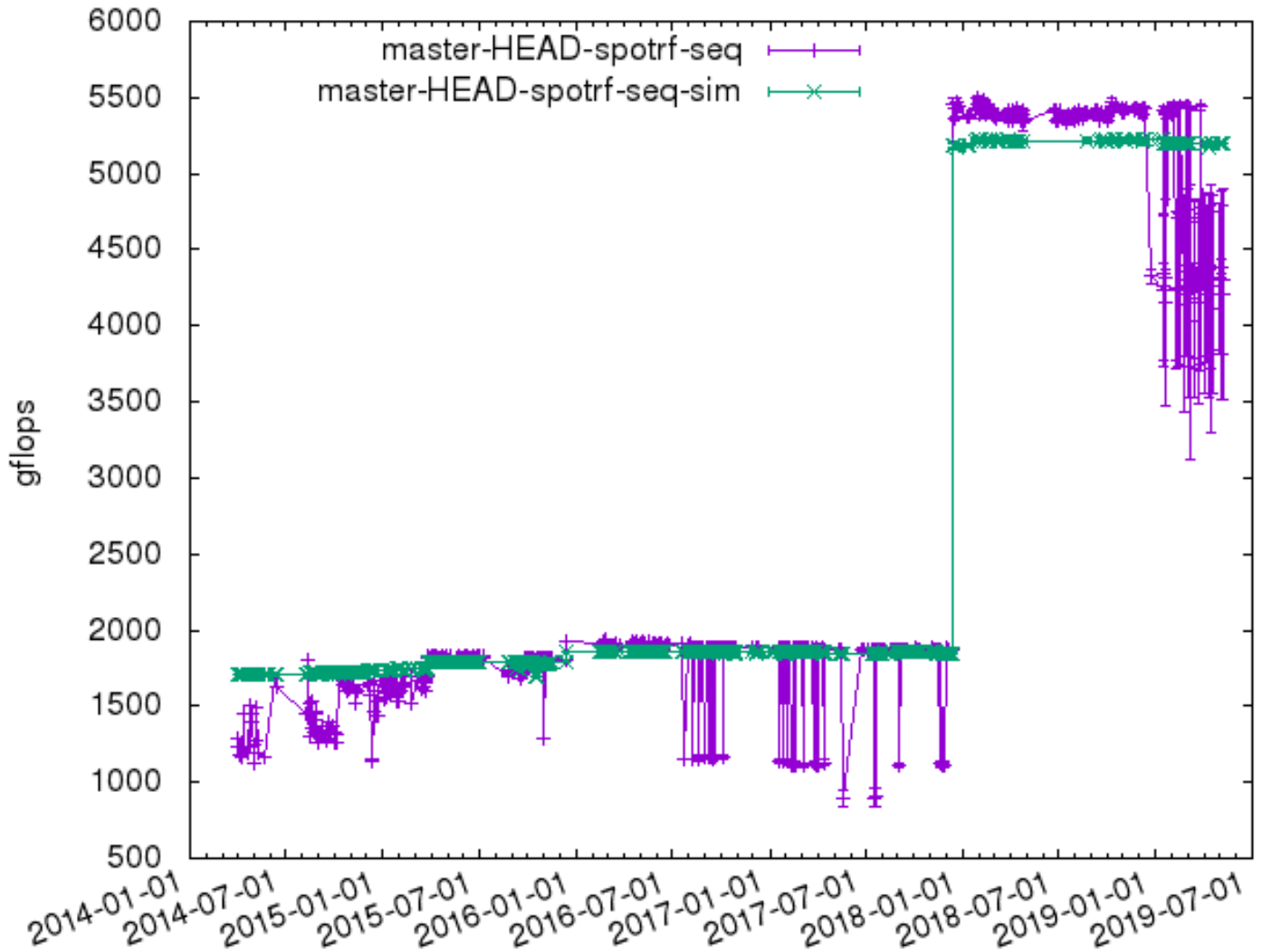
Execution behavior is representative

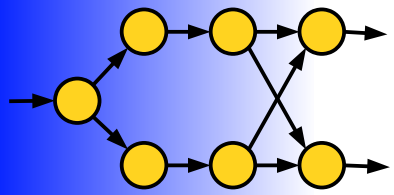




Results

More trustworthy Continuous Integration regression testing

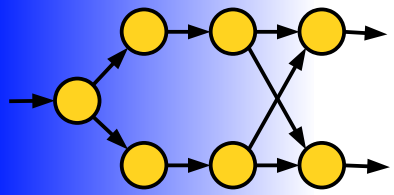




Research opportunities

Experimenting scheduling heuristics

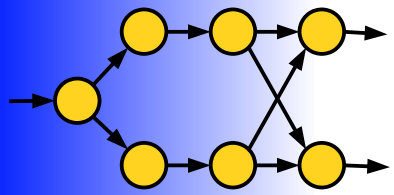
- With representative & reproducible simulated results
  - Confidence
- On various target platforms
  - Different GPU models
  - Different PCI topologies
  - …
  - Without any platform account or CPU.hour allocation



Research opportunities

Experimenting scheduling heuristics

- With tuned platform
  - Change GPU memory size
  - Change PCI bus speed
  - Change topology (e.g. NVLink)
- With invented platforms
  - Assemble existing performance models
    - Different kinds of GPUs in same system
  - Or even write some
- Could also be used for provisioning
  - “Does this application, with 8 V100 GPUs, needs NVLink?”

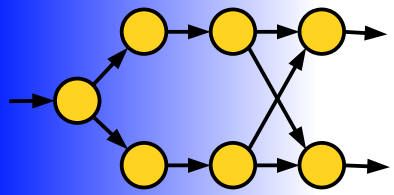


Research opportunities

Experimenting scheduling heuristics

- Ignoring data transfer costs
  - Set PCI bandwidth to infinity
  - Scheduling tasks without caring about locality, as first step
- Ignoring scheduler cost
  - No need to care about efficient implementation, as first step

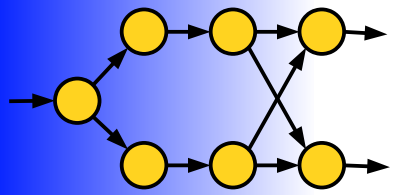
- Suraj Kumar's PhD thesis mostly in simulation
  - Scheduling improvements then confirmed with measurements



Research opportunities

Debugging scheduling heuristics

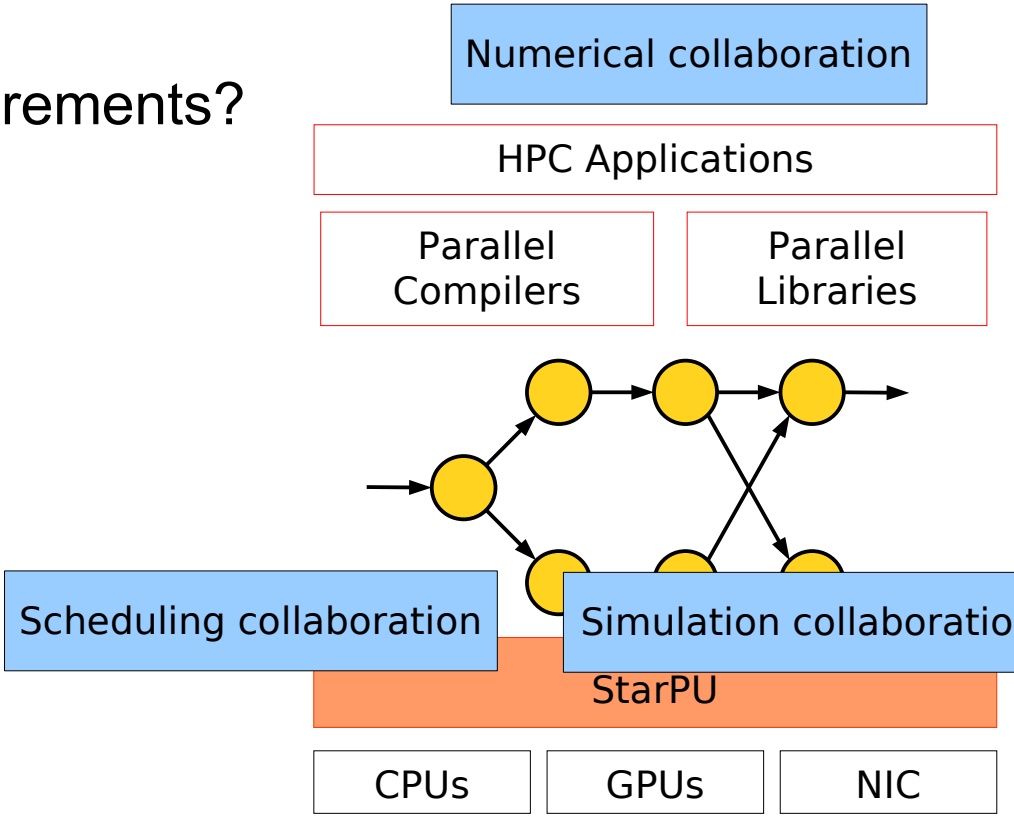
- Deterministic bugs
  - Can reproduce them at will
  - Adding printfs do not change behavior
- Breakpoint on timestamp
  - Inspect runtime system state etc.
  - Direct link between artifact in trace and scheduling decision



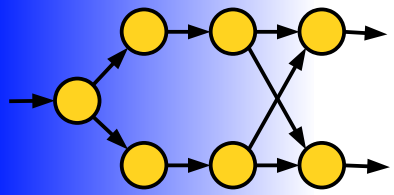
Conclusion

Simulation of a runtime system

- Trustworthy results
  - Even more than actual measurements?
- Playground for scheduling research
  - Reproducibility
  - Various testcases
  - Progressive exposition of theory to practice
- Not discussed today
  - MPI support
  - Irregular kernel modeling



http://starpu.gforge.inria.fr/



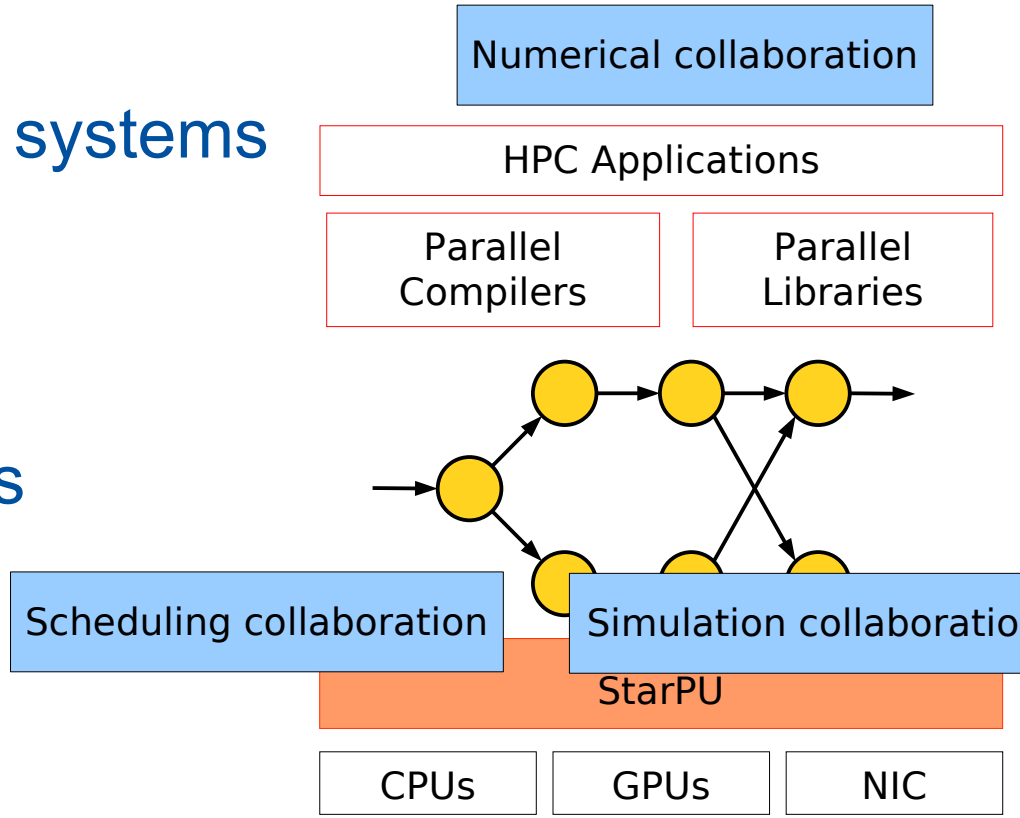
Perspectives

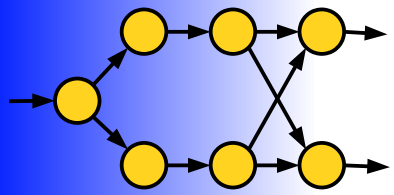
Task graph market

- Collecting application cases
- Tools to assemble synthetic systems

Refined CPU kernel modeling

- Cache effects between tasks
  - Scheduling decision thus has impact
  - Germán Ceballos PhD
- NUMA systems
  - Isolation of cache / PCI bus effect
  - Idriss Daoudi PhD

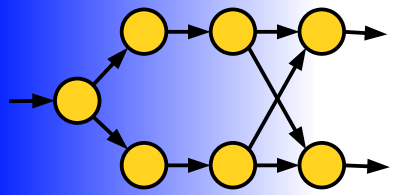


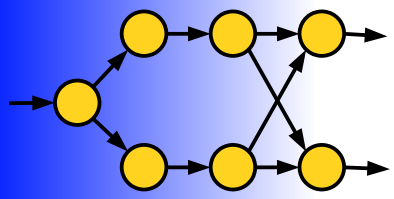


Thanks!

[AugonnetPhD11] [ArrasPhD15] [RossignonPhD15] [SergentPhD16] [KumarPhD17] [LionPhD]

- ANR ProHMPT'09, MEDIAGPU'10, FP3C'10, SONGS'12, SOLHAR'13
- IPL HAC-SPECIS'16, HPC-BigData'18
- EU PEPPHER'10, EXA2PRO'18
- Rapid Hi-BOX'13
- MORSE associate-team
- Used by AL4SAN, Chameleon, ExaGeoStat, FLUSEPA, HiCMA, hmat, KSVD, MAGMA, MaPHyS, MOAO, PaStiX, QDWH, qr_mumps, ScalFMM, SCHNAPS, SignalPU, SkePU, STARS-H





Performance models - MultiLinear

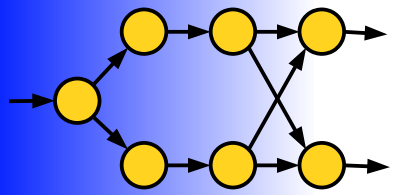
- Theoretical complexity of BLAS GEQRT

$$T_{\texttt{GEQRT}} = a + 2b(NB^2 \times MB) - 2c(NB^3 \times BK) + \frac{4d}{3}NB^3$$

- Can tune a, b, c, d with linear regression
- Why these NB^2xMB, NB^3xBK and NB^3 parameters ?

	GEQRT Duration	
Constant	-2.49×10^1 $(-2.83 \times 10^1, -2.14 \times 10^1)$	***
$NB^2 \times MB$	5.49×10^{-7} $(5.46 \times 10^{-7}, 5.51 \times 10^{-7})$	***
$NB^3 \times BK$	-5.52×10^{-7} $(-5.57 \times 10^{-7}, -5.48 \times 10^{-7})$	***
NB^3	1.50×10^{-5} $(1.30 \times 10^{-5}, 1.70 \times 10^{-5})$	***
Observations	493	
R^2	0.999	
Note:	*p<0.1; **p<0.05; ***p<0.01	

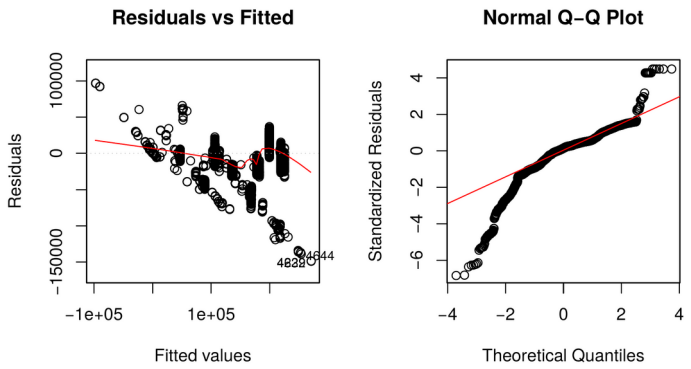
- Works well with regular kernels



Performance models - ScalFMM

- Initial attempt with direct parameters

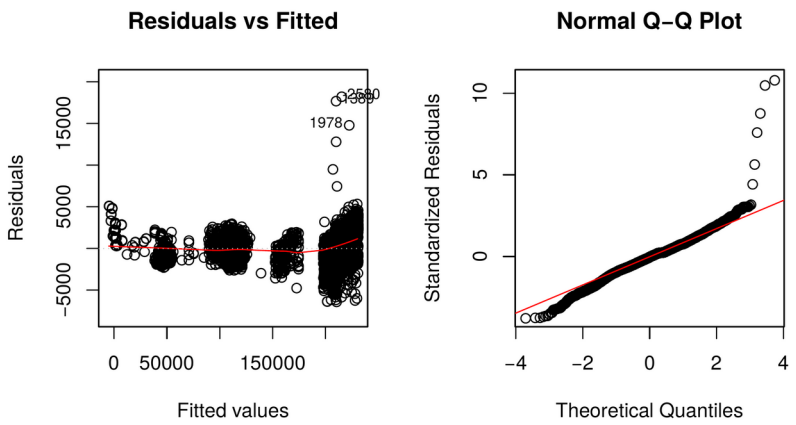
$TreeLevel$	7.73×10^{1} $(7.32 \times 10^{1}, 8.14 \times 10^{1})$	***
$NbCells$	1.52×10^{2} $(1.46 \times 10^{2}, 1.59 \times 10^{2})$	***
$IntervalSize$	-5.55×10^{-1} $(-6.65 \times 10^{-1}, -4.45 \times 10^{-1})$	***
$TreeLevel^2$	-6.73 $(-7.10, -6.37)$	***
$NbCells^2$	-6.08×10^{-3} $(-7.94 \times 10^{-3}, -4.21 \times 10^{-3})$	***
$IntervalSize^2$	-3.63×10^{-12} $(-6.36 \times 10^{-12}, -9.04 \times 10^{-13})$	
$TreeLevel \times NbCells$	-4.68 $(-5.88, -3.48)$	***
$TreeLevel \times IntervalSize$	6.24×10^{-2} $(5.02 \times 10^{-2}, 7.46 \times 10^{-2})$	***
$NbCells \times IntervalSize$	1.81×10^{-4} $(1.39 \times 10^{-4}, 2.24 \times 10^{-4})$	***
$TreeLevel \times NbCells \times IntervalSize$	-2.08×10^{-5} $(-2.55 \times 10^{-5}, -1.61 \times 10^{-5})$	***
Constant	-2.27×10^{2} $(-2.40 \times 10^{2}, -2.14 \times 10^{2})$	***
Observations	4,987	
R^2	0.906	
Note:		*p<0.1; **p<0.05; ***p<0.01

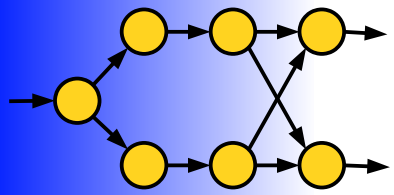


Performance models - ScalFMM

- Introduce NbInteraction parameter

$TreeLevel$	9.10×10^2 $(8.46 \times 10^2, 9.74 \times 10^2)$	***
$NbCells$	5.34 $(5.17, 5.50)$	***
$NbInteractions$	8.59×10^{-1} $(8.58 \times 10^{-1}, 8.60 \times 10^{-1})$	***
Constant	-7.09 $(-7.50, -6.67)$	***
Observations		4,987
R^2		0.999
Note:		*p<0.1; **p<0.05; ***p<0.01

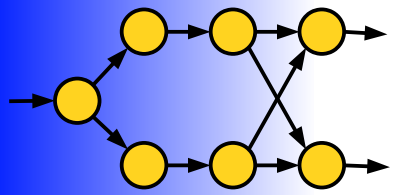




Performance models - ScalFMM

Performance models - ScalFMM





Performance models - ScalFMM