# SimGrid @ DarkEra


Martin Quinson, Frédéric Suter
(and the SimGrid team)

June 1., 2021

## Agenda

- Modern large-scale distributed systems
- Simulating distributed systems
- Introduction to SimGrid
- DarkEra: Motivation and challenges
- Hands on! and/or discussion

# Modern Large Scale Distributed Systems

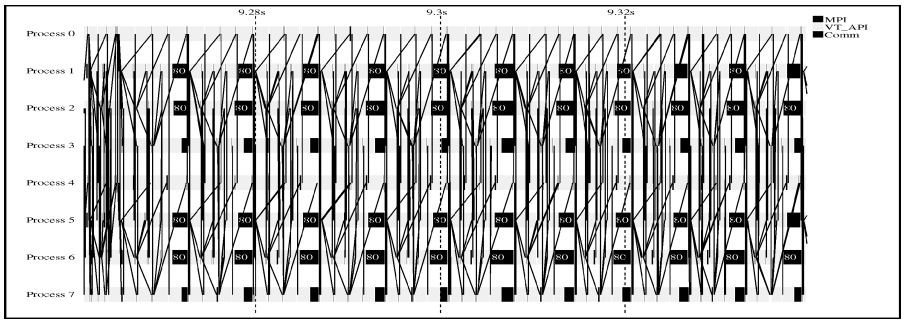
## HPC: Huge regular systems

#1 Fugaku (2020)
7,630,848 cores
158,976 x (48+4 ARM cores)
538 Pflops, 30MW

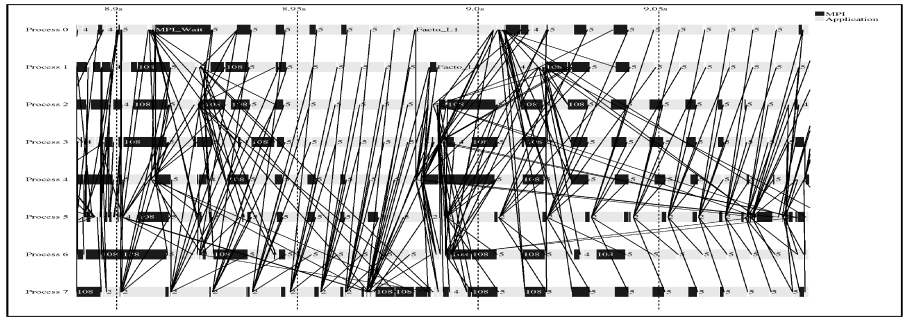

#2 Summit (2018)
2,282,544 cores
4608×(2×22-cores + 6GPU)
122 Tflops, 9MW


#4 Taihu Light (2016)
10,649,600 cores
40 960× 260-cores RISCs
93 Tflops, 15MW

## Applications complexity is raising



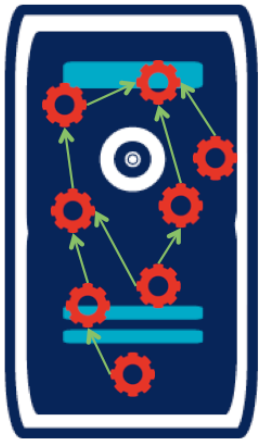

Rigid, Regular, Hand-tuned Comm Patterns



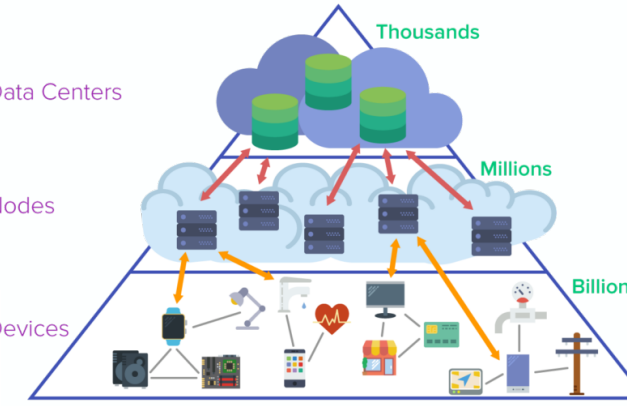

Dynamic, Irregular (task-based?)

How do we study these beasts?

# Cloud Computing, IoT, SmartCity, Fog, ...

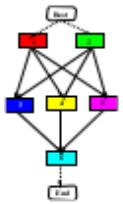




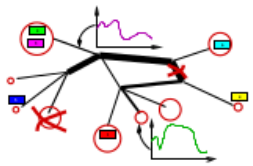


# Simulating Distributed Systems

## Fastest Path from Idea to Data



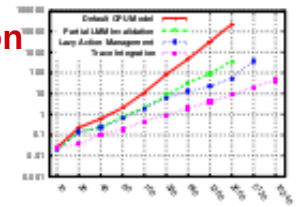
Idea or
MPI code



Experimental
Setup

Simulation

Models

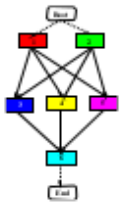


Scientific Results

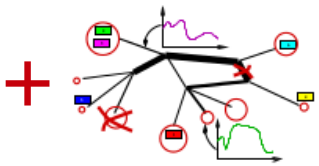
- Centralized, Reproducible, Clairevoyance, no Heisenbug, *What if* studies

# Simulating Distributed Systems

Fastest Path from Idea to Data or even Study of Real Systems



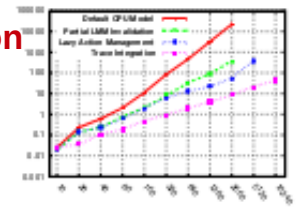
Idea or MPI code



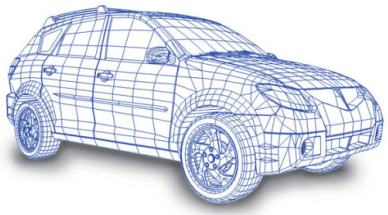
Experimental Setup

Simulation

Models



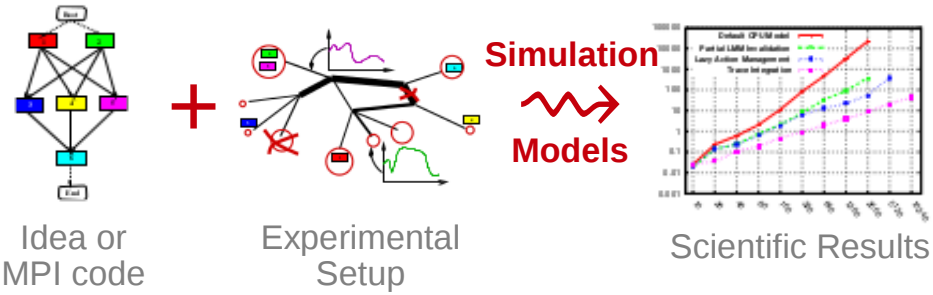
Scientific Results

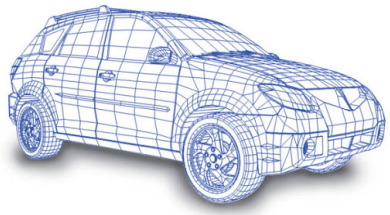


- Centralized, Reproducible, Clairevoyance, no Heisenbug, *What if* studies
- Co-design, capacity planning and/or hardware qualification

# Simulating Distributed Systems

## Fastest Path from Idea to Data or even Study of Real Systems



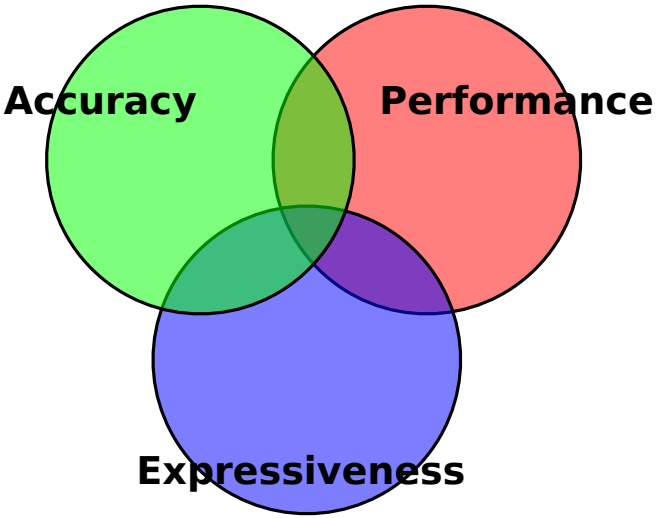



- Centralized, Reproducible, Clairevoyance, no Heisenbug, *What if* studies
- Co-design, capacity planning and/or hardware qualification

## Simulator Expected Qualities

- Expressiveness: Capture right concepts (specific vs. generic)
- Accuracy: Validated Results (vs scalability?); QoS (tool vs. prototype)
- Performance (size & speed); Usability: Don't fool the users, documentation

# State of the Art




## HPC: Performance is everything

- Accuracy matters for performance evaluation
- Performance: DES vs. PDES; Online vs. Offline
- Accuracy: Coarse- vs. Fine-grain models
- Expressiveness often limited to MPI

## Cloud: Many expressive simulators

- Great for quick studies with poor accuracy
- Narrow focus, short-lived research prototypes
- CloudSim, DartCSim+, iFogSim, etc.

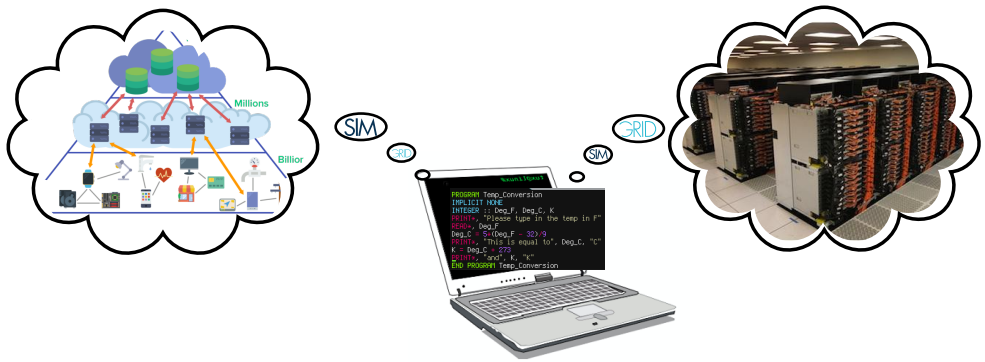
## SimGrid focuses on accuracy, versatility and performance

- Validated models (time, energy) of CPU, networks (TCP, wifi), disks
- Expressiveness: high versatility, but provides low-level building blocks
- Historically, more HPC but now decent support for Cloud/Fog/IoT

# SimGrid: Versatile Simulator of Distributed Apps

## Install a Scientific Instrument on your Laptop

- Joint Project since 1998, mostly from French institutions
- Open Project: users around the world (many academic and some industrial)
  - Almost 2000 publications only cite it, 60+ extend it, 400+ use it
  - Grid, Cluster, P2P, Volunteer computing, Hadoop, HPC, Cloud, Fog ...
- Several projects and extensions: Wrench, BatSim; StarPU, BigDFT; ProxyApp




## Key Strengths

- Usability: Fast, Reliable, User-oriented APIs, Visualization
- Validated performance models Open Science $\rightsquigarrow$ Predictive Power
- Architectured as an OS $\rightsquigarrow$ Efficiency; Performance & Correction co-evaluation
- Versatility: Advances in HPC modeling reused by Clouds users

# An MPI example

## The application

```
#include <mpi.h>
int main(int argc, char**argv) {
  int x;

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &x);

  if (x == 0) { // rank
    x = 42;
    MPI_Send(&x, 1, MPI_INT, 1, 1,
             MPI_COMM_WORLD);
  } else {
    MPI_Recv(&x, 1, MPI_INT, 0, 1,
             MPI_COMM_WORLD, NULL);
    printf("Got %d from rank 0",x);
  }
  MPI_Finalize();
}
```

## hostfile.txt

```
node-0.acme
node-1.acme
```

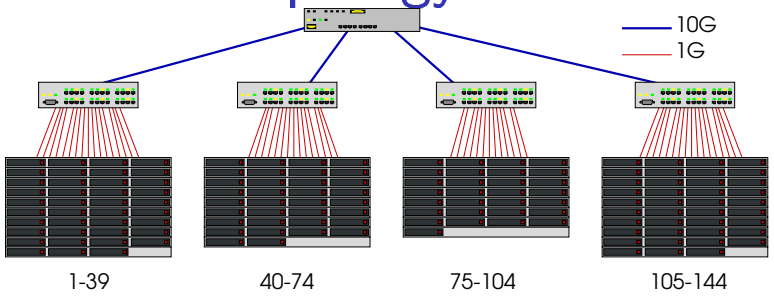
## XML Platform File

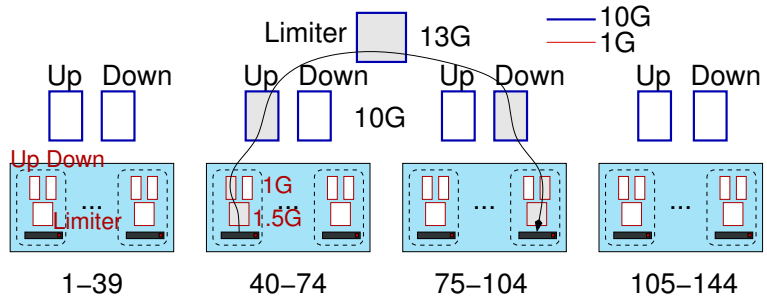
```
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM
"http://simgrid.gforge.inria.fr/simgrid.dtd">
<platform version="4">
<cluster id="acme"
 prefix="id-"  radical="0-9" suffix=".acme"
 power="1Gf"   bw="125MBps"  lat="50us"
 bb_bw="2GBps" bb_lat="500us"/>
</platform>
```

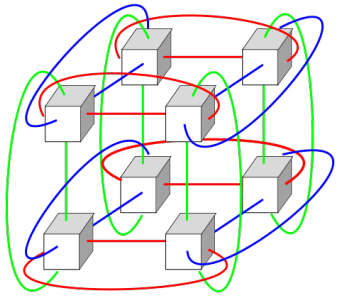
```
$ smpicc source.c -o application  # The code is now compiled
$ smpirun -platform cluster.xml -hostfile hostfile.txt  ./application # It starts
[...]  # Some debug information about your data provenance
Got 42 from rank 0
```

# Define and Calibrate Platforms (1/2)

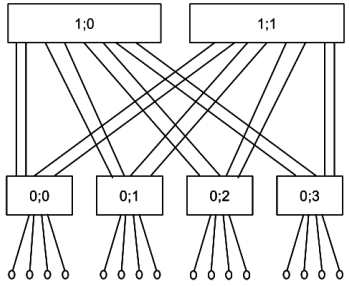
## Network topology and contention



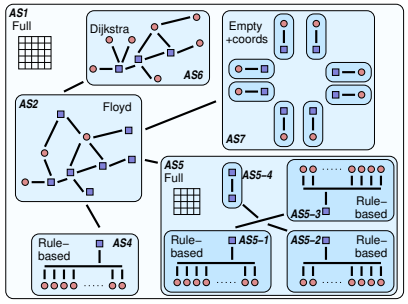



Torus

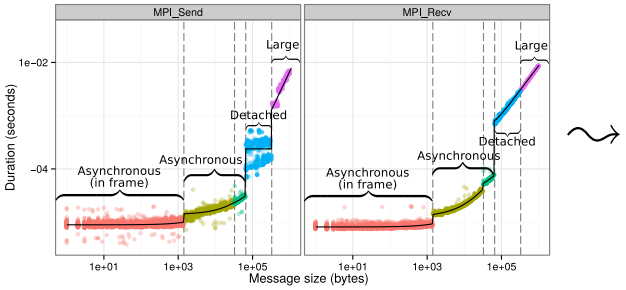



Fat-trees

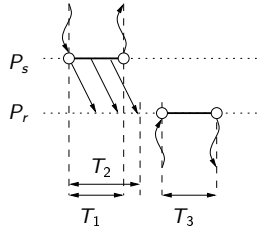



Hierarchies of ASes

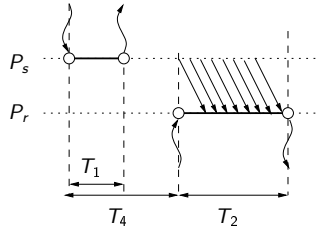
## Hybrid point-to-point communication model



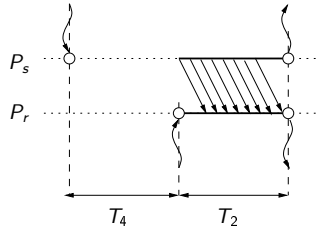

Asynchronous ($k \leqslant S_a$)




Detached ($S_a < k \leqslant S_d$)




Synchronous ($k > S_d$)



# Define and Calibrate Platforms (2/2)

## Thresholds for modes

```
<prop id="smpi/async_small_thres" value="65536"/>
<prop id="smpi/send_is_detached_thres" value="327680"/>
```

## Factors for latency and bandwidth

- For various message sizes, computed from regression

```
<prop id="smpi/bw_factor" value="size1:x;size2:y ..."/>
<prop id="smpi/lat_factor" value="size1:x;size2:y ..."/>
```

## Timings to inject in asynchronous operations

```
<prop id="smpi/os" value="size1:x1:x2;size2:y1:y2 ..."/>
<prop id="smpi/ois" value="size1:x1:x2;size2:y1:y2 ..."/>
<prop id="smpi/or" value="size1:x1:x2;size2:y1:y2 ..."/>
```

http://simgrid.gforge.inria.fr/contrib/smpi-calibration-doc/

# SimGrid 4: the S4U interface

## Modern interface

- Generic and consistent API
- C++14 core, with C and Python bindings

## Simple concepts

- Actors: run user code ($\approx$ thread or processes)
- Activities: explicit communication, computation, disk usage, synchro
- Resources: Host, Links, Disks. Shared through performance models.
- Scalable platforms: Hierarchy of network zones, with user-defined routing

## Extensible through plugins

- Attach code to signals fired automatically by the simulation kernel
  - Actor: `creation`, `suspend`, `resume`, `sleep`, `wakeup`, `migration`, `termination`
  - Comm, Exec, IO, Synchro: `creation`, `start`, `state_change`, `completion`
  - Host, Link, Disk: `creation`, `state_change`, `speed_change`, `destruction`
  - Engine: `platform_creation`, `time_advance`, `simulation_end`, `deadlock`
  - VM: `start`, `suspend`, `resume`, `migration`, `termination`, NetZone
- Extend simulation objects with your data: Activities, Resources, NetZones

# Existing plugins in SimGrid 4

## Simulation capability

- VM migration: live in 2 phases, with dirty page tracking (default: immediate)
- File system: file creation/rm; disk capacity (default: only read/write activities)

## Simulation models

- VM sharing, DVFS: Implemented through plugins
- Resource load tracking: adapt this plugin to your scheduling algorithm
- CPU and link energy consumption: 200 lines each

## Planned/possible models

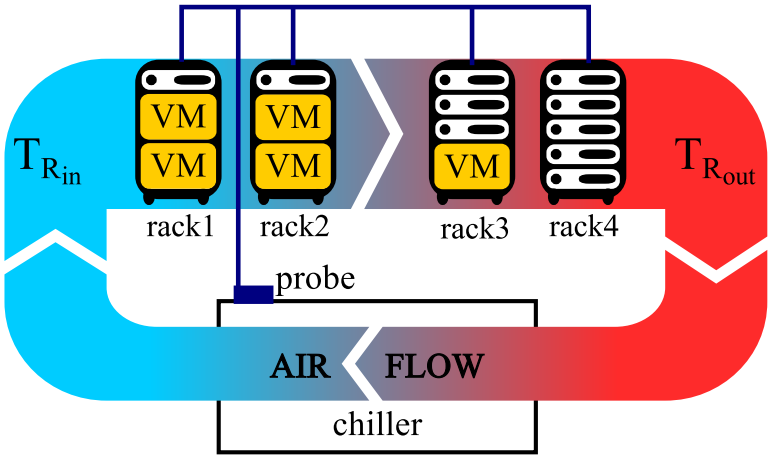
- Other VM migration model
- Power capping: reducing computing performance when heating
- ...

# Co-simulation with SimGrid

## Packet-level simulation

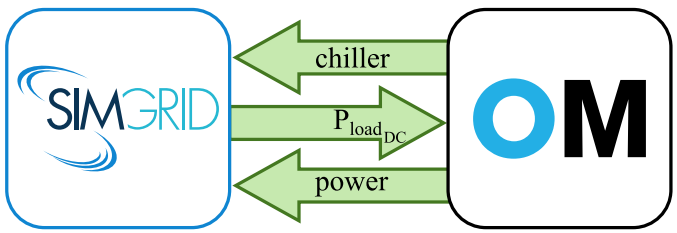
- ns-3 can be plugged in to replace our models (to validate them)

## Cyberphysical systems: A Data Center with a chilling facility




- Servers host VMs, which load heats the air
- The chiller cools the air flow, up to a point
- Above a given load threshold, the chiller stops
- Above a given temp., the DC shuts down
- Q: How to migrate the VMs before shutdown?

- Server heating + Cooling in OpenModelica
- IT system as SimGrid actors
- Server load: SG→ OM; Temp prob: OM→ SG

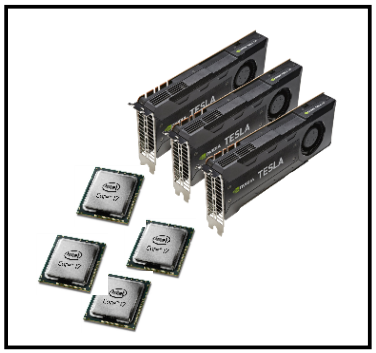



## SimGrid-FMU: generic co-simulation (import 2.0 FMUs)

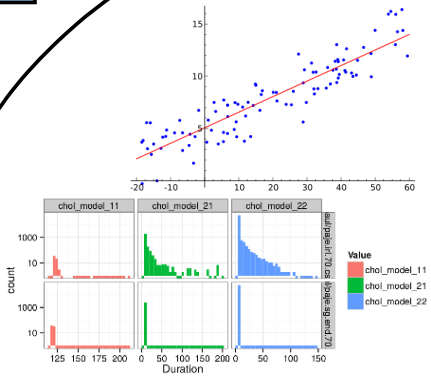
- Implemented as a plugin on top of SimGrid, released separately
- Used internally to interface with Power Factory and Power Panda

# Simulation of GPU for StarPU

## Calibration



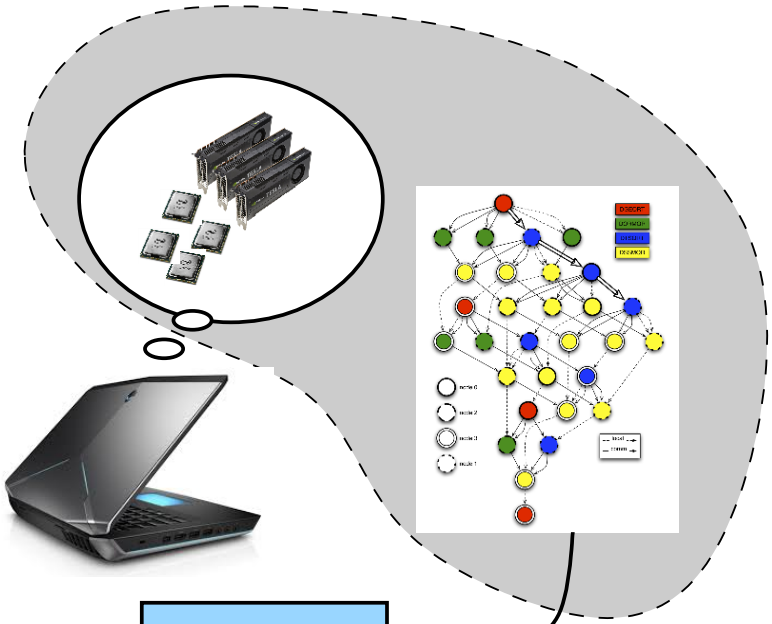
StarPU




Performance Profile

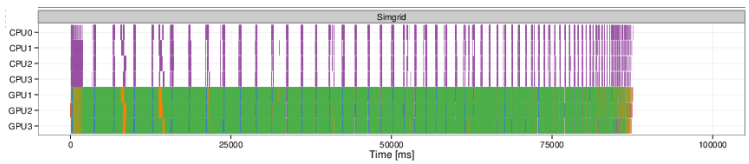
Run once!

## Simulation



StarPU

SimGrid

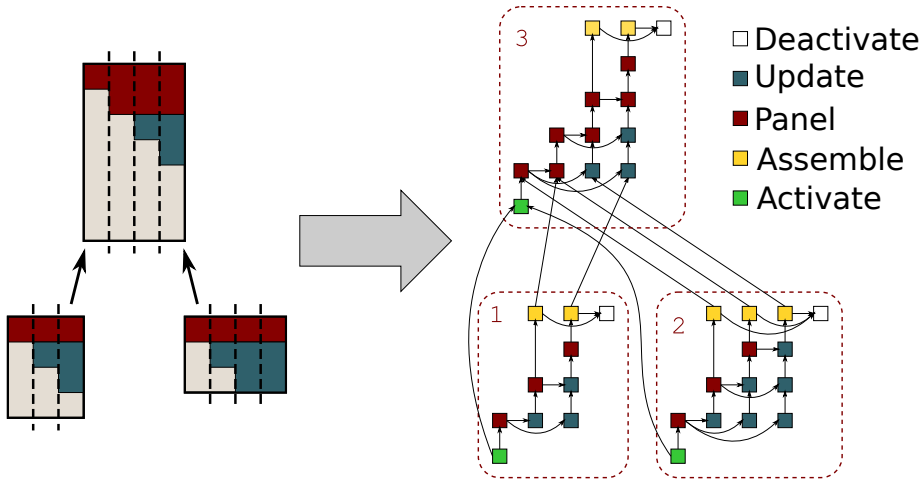


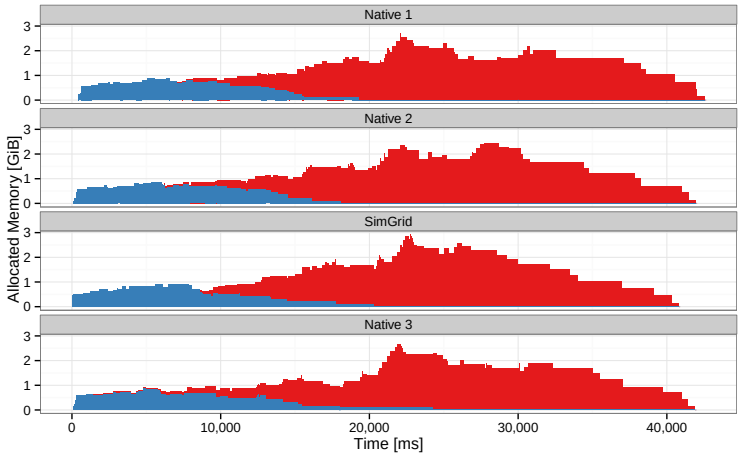

Quickly Simulate Many Times

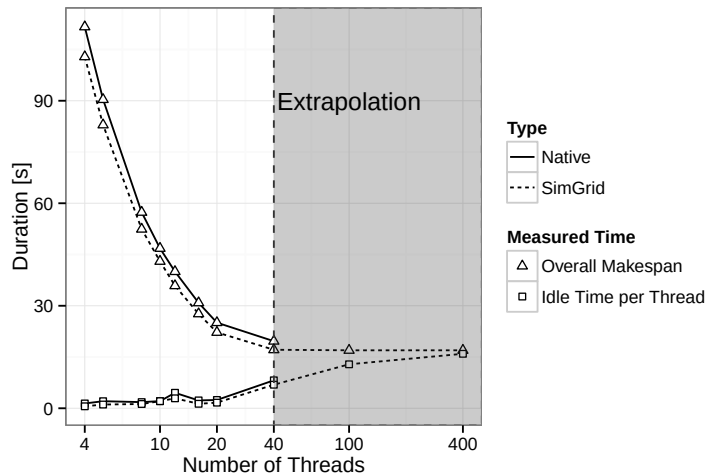
# StarPU QR-Mumps

## QR-MUMPS multi-frontal sparse factorization on top of StarPU

- Tree parallelism
- Node parallelism
- Variable matrix geometry
- Fully dynamic scheduling w. StarPU



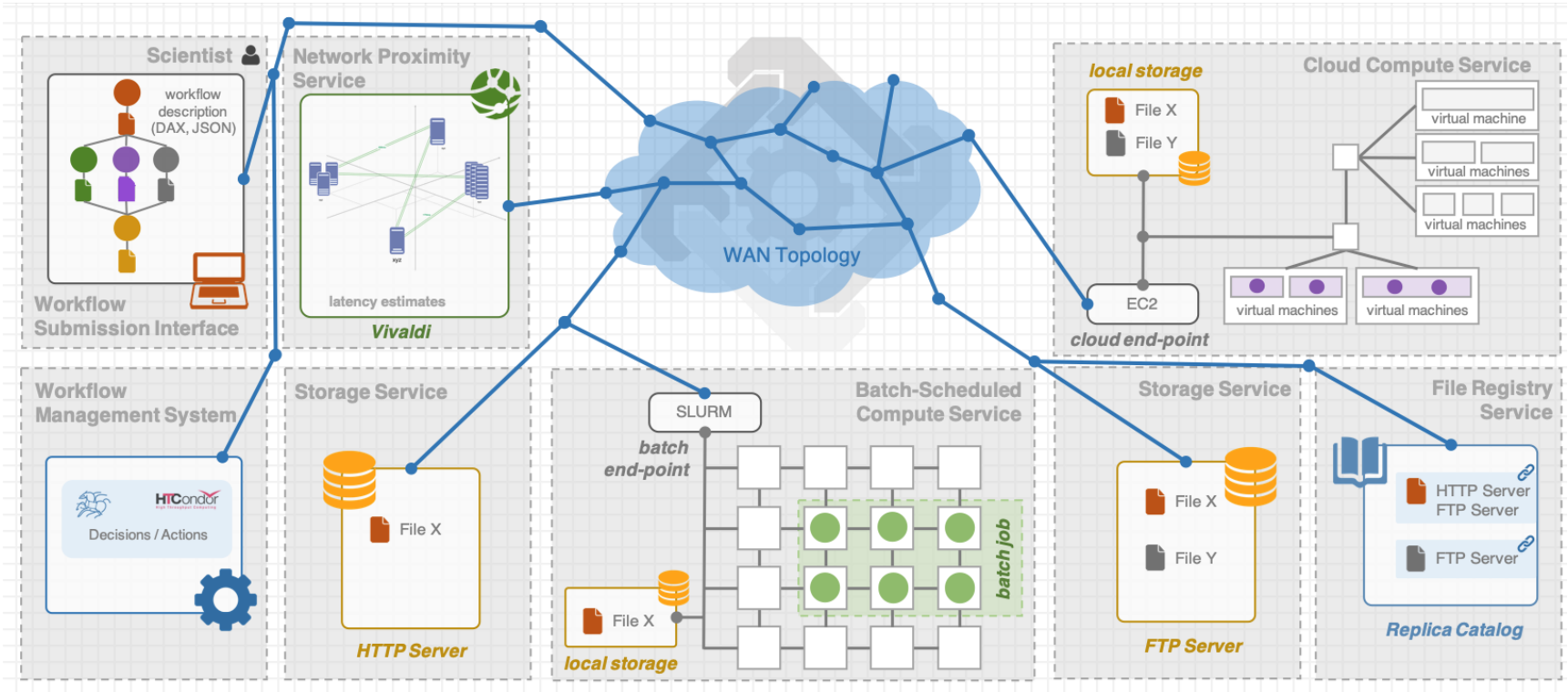




## DarkEra Challenge

- Co-simulation with PREESM: calibration without direct benchmarking

# Wrench: Capable simulator on top of SimGrid

- High-level building blocks for developing custom simulators
  - Computation (Cloud EC2-like, Batch, Rack), Storage (FTP, HTTP, P2P)
  - Monitoring (Vivaldi, Sonar), Replica catalogs, etc
- Simulated production systems: WMS (Pegasus, Moteur, Makeflow), Hadoop
- Based on SimGrid models for accuracy and performance; Improved usability




DarkEra challenge: Dataflow programming model; specific high level prototyping

# Conclusion

## SimGrid: accurate and scalable low-level building blocks (since 1998)

- Efficient core implemented in C++14, with Python bindings (among other)
- Performance models (time, energy) for CPU, network (wired, wifi) and disk
- Extensive testing work; now with a documentation and several tutorials

## Co-simulation with SimGrid

- Packet-level networks with ns-3 (GTNets)
- Generic co-simulation with FMU: Modelica, Power Factory, Power Panda

## Wrench: high-level blocks to simulate Cluster/Cloud systems

- Declare a full platform in few lines, get comprehensive simulation vizualizations
- Ready to assemble bricks (two abstraction levels: batch queue vs full Hadoop)

## SimGrid @ DarkEra

- Co-simulation with PREESM for what-if calibration of compute accelerators
- Allow quick prototyping for SKA computing (Dataflow programming model etc)