Multi-resource planning: Simulations and study of a new scheduling approach for distributed data production in High Energy and Nuclear Physics

Dzmitry Makatun [1 3] Jérôme Lauret[2]
Hana Rudová [4] Michal Šumbera [1]

[1]Nuclear Physics Institute, Academy of Sciences, Czech Republic

[2]Brookhaven National Laboratory, USA

[3]Czech Technical University in Prague, Czech Republic

[4]Faculty of Informatics, Masaryk University, Czech Republic

makatun@rcf.rhic.bnl.gov

Outline

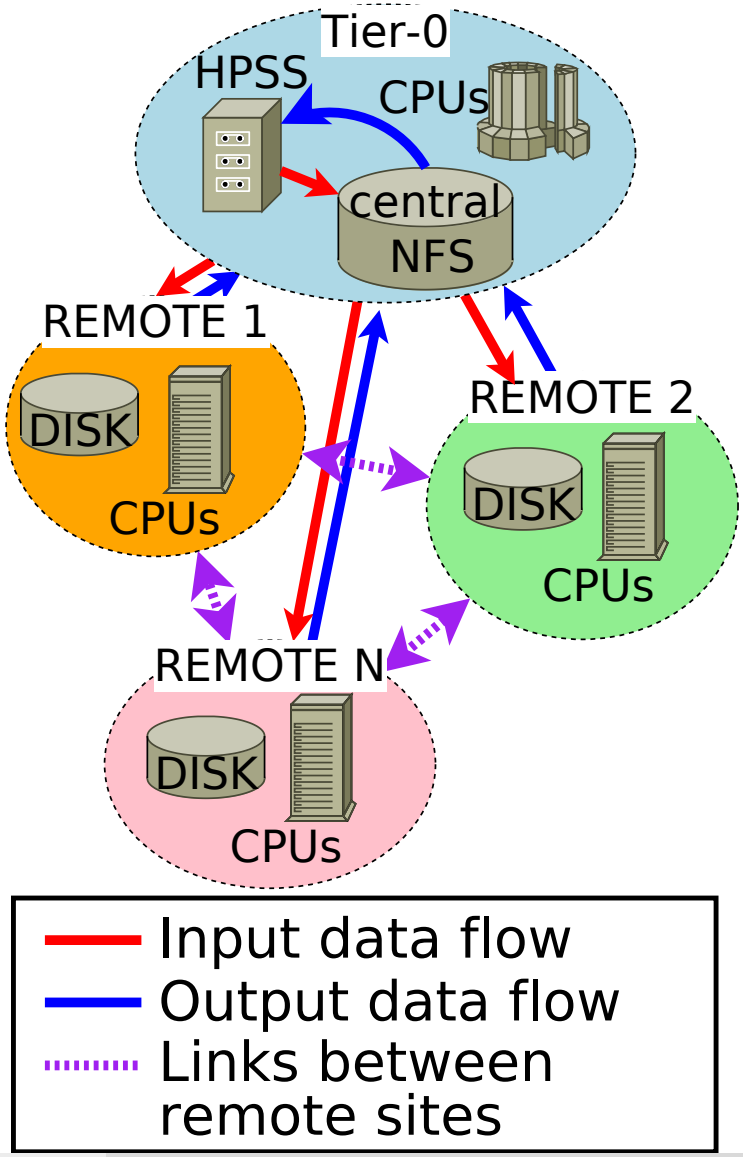
Distributed data production

Used properties

- Single central storage (source of input & destination for output data)
- Job: 1 input and 1 output files, 1 CPU, processed once
- Output size $\lesssim$ input size $\sim$ duration

Planning remote site usage

- Computational resources are available at Tier-0 and remote sites (Tier-1s, Tier-2s ...).
- **How should we distribute a given set of files between sites to complete the processing faster?** consider: {CPUs, disk, bandwidth}
- Real network topology: shared links, links between remote sites. **Which transfer path?**



Network flow model

Idea

Plan resource load only and then distribute particular jobs accordingly

Knowing data distribution and status of comp. sites at the start time, **how much data should be transferred over each network link?**

Model basics

- ΔT - planning time interval (e.g., 12 hours)
- $Link\ weight = bandwidth \cdot \Delta T$
- $Flow_{link}$ - amount of data to be transferred in ΔT

Sub-problems:

- Input
- Output

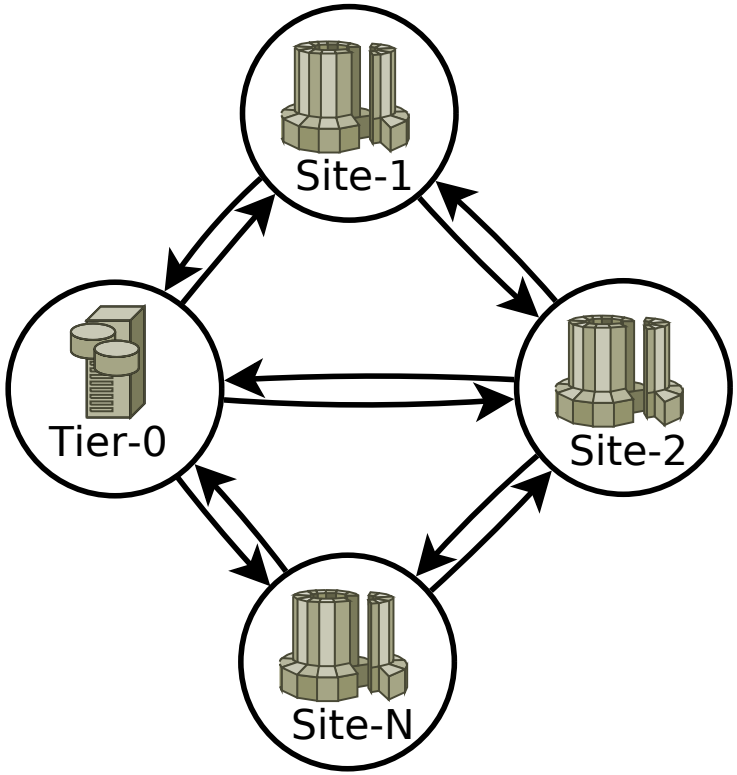
Repeat planning at the beginning of every ΔT

Network flow maximization problem has polynomial complexity (good)

Network flow model

step 1

Grid as a directed weighted graph

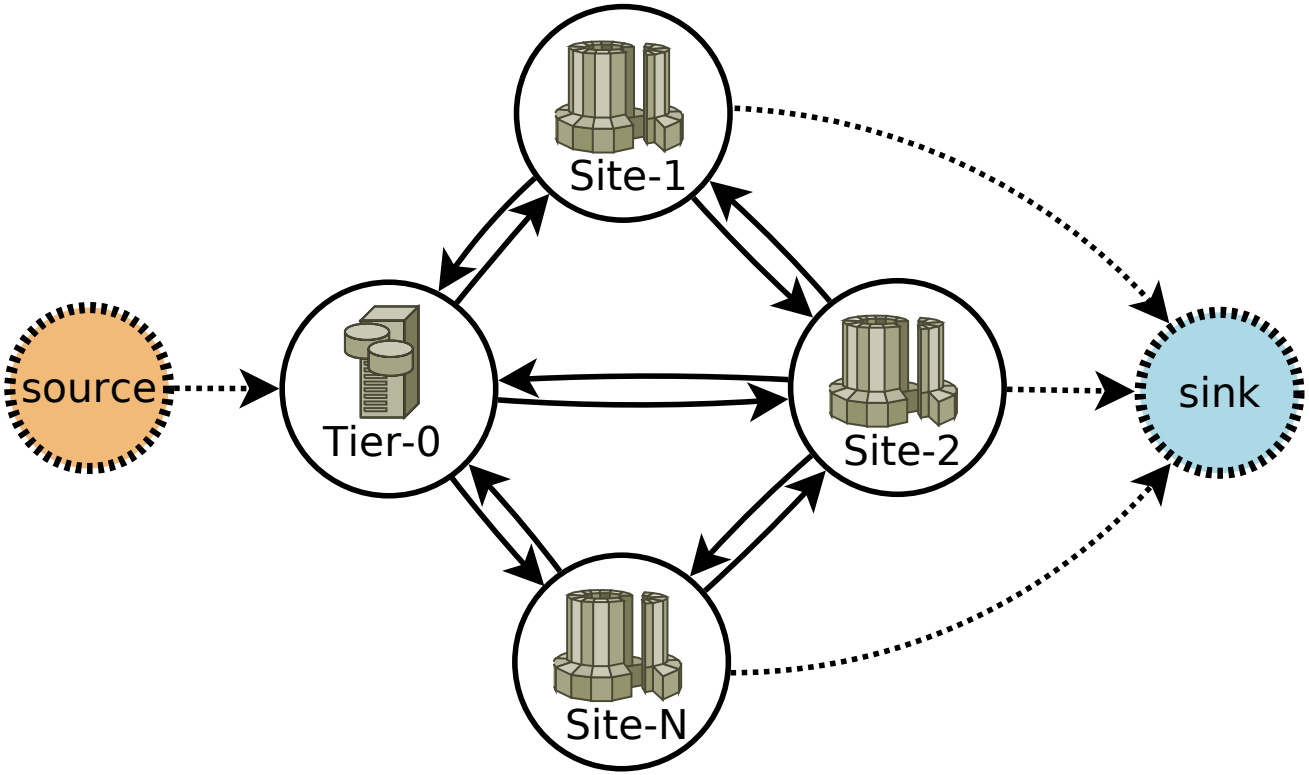


$$Link\ weight = bandwidth \cdot \Delta T$$

Network flow model

step 2

Network for the **input** problem

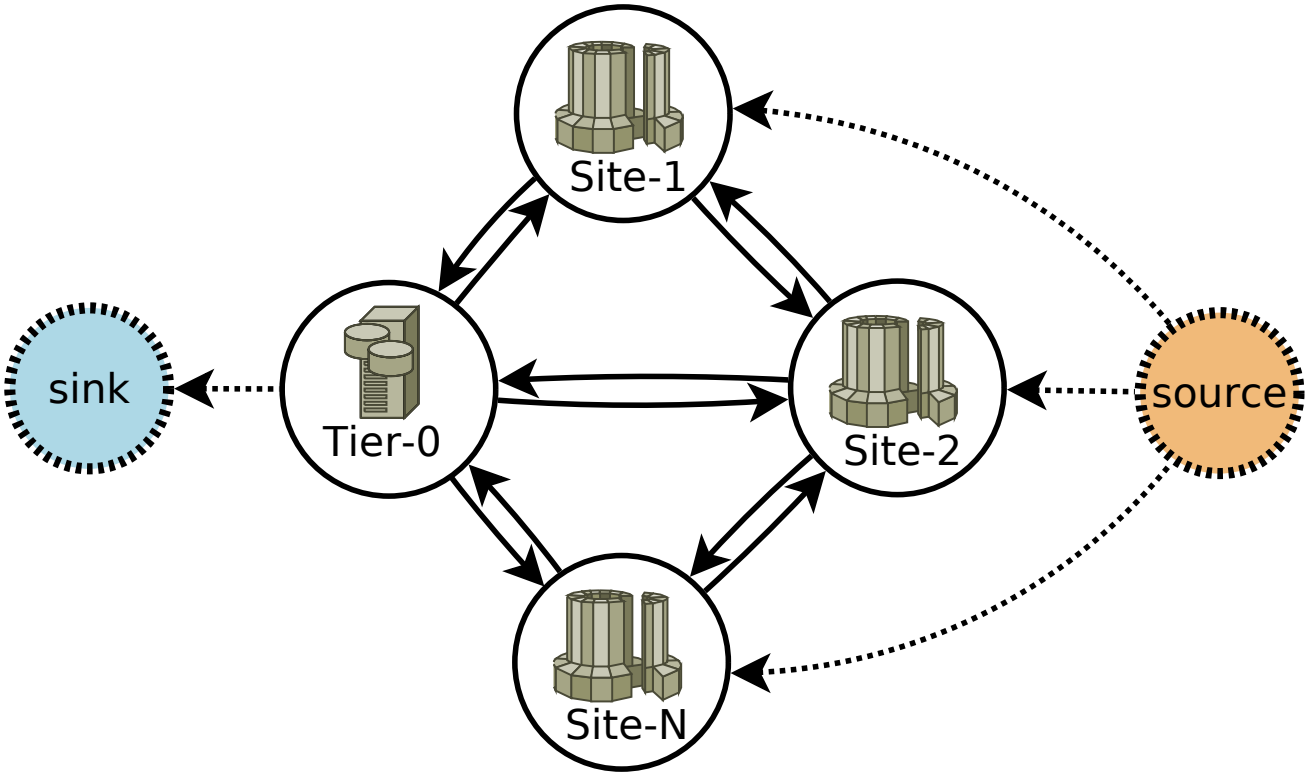


Dummy edges: constraints on storage and CPUs at each site.

Network flow model

step 3

Network for the **output** problem



Dummy edges: available (& expected) output data.

Plan execution

Handler

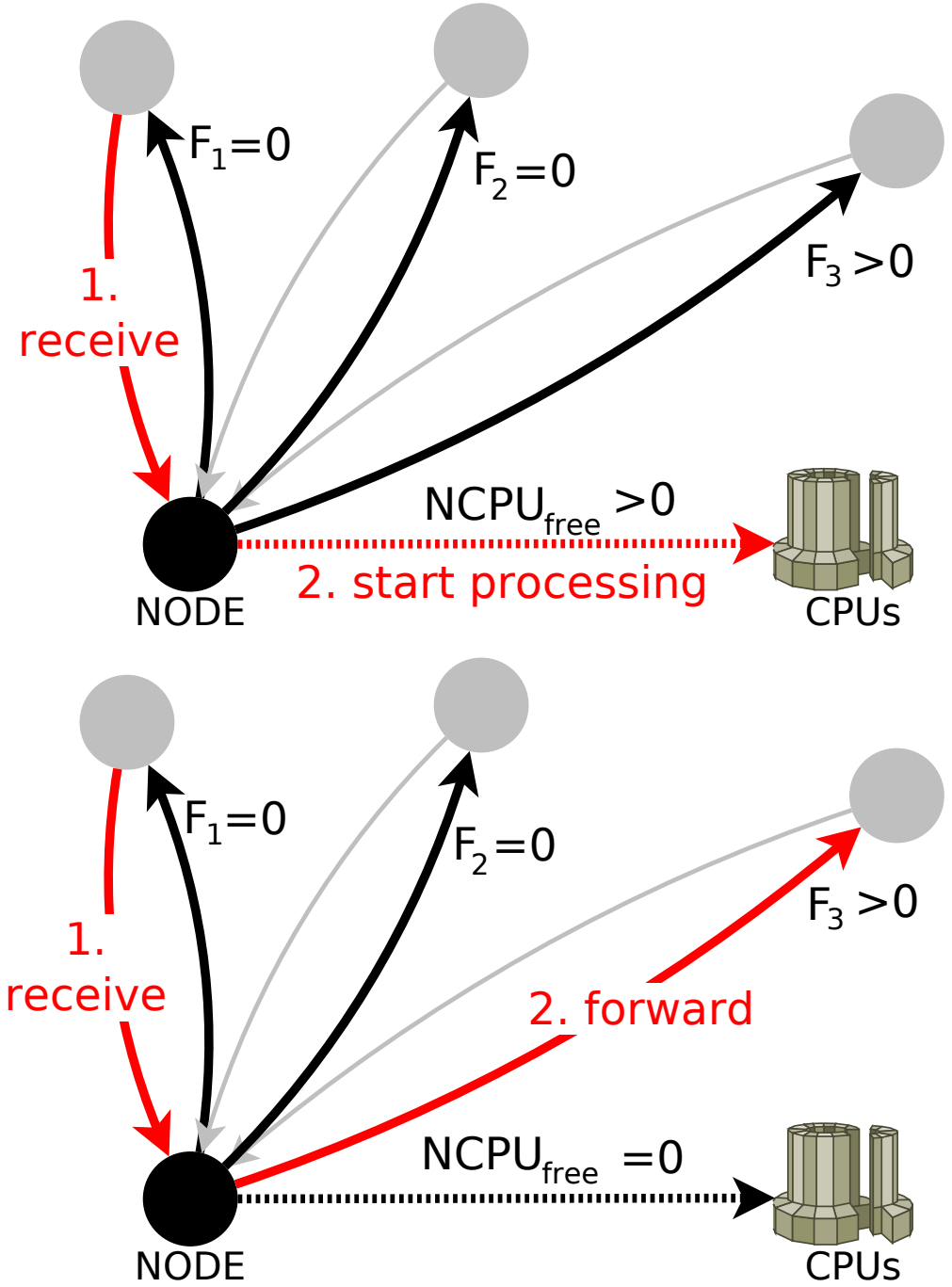
- Service runs at each site
- Sends status data
- Executes the plan

When a new file arrives:

Process the file (if $NCPU_{free} > 0$)

OR Forward it over one of the links (if $\exists F_l > 0$), decrease F_l

OR Keep the file until a new plan or a free CPU appears



Simulations using GridSim

GridSim [Buyya and Murshed, 2002]

- A toolkit for simulation of computational Grid (java library)
- Discrete event simulation
- Elements: resource, machine, CPU, storage, network link, router, packet, job, file, user
- Data exchange between entities
- Models for job to CPU scheduling, file transfer, etc.

Jobs from monitoring database [Hajdu et al., 2016] (poster #5)

- STAR @ KISTI
- 7 months in 2014
- ∼77,000 jobs
- Can generate larger sets using MC selection

Simulated scheduling approaches

File transfer modes

- **Sequential**: 1 transfer at a time
- **Parallel**: multiple transfers share bandwidth

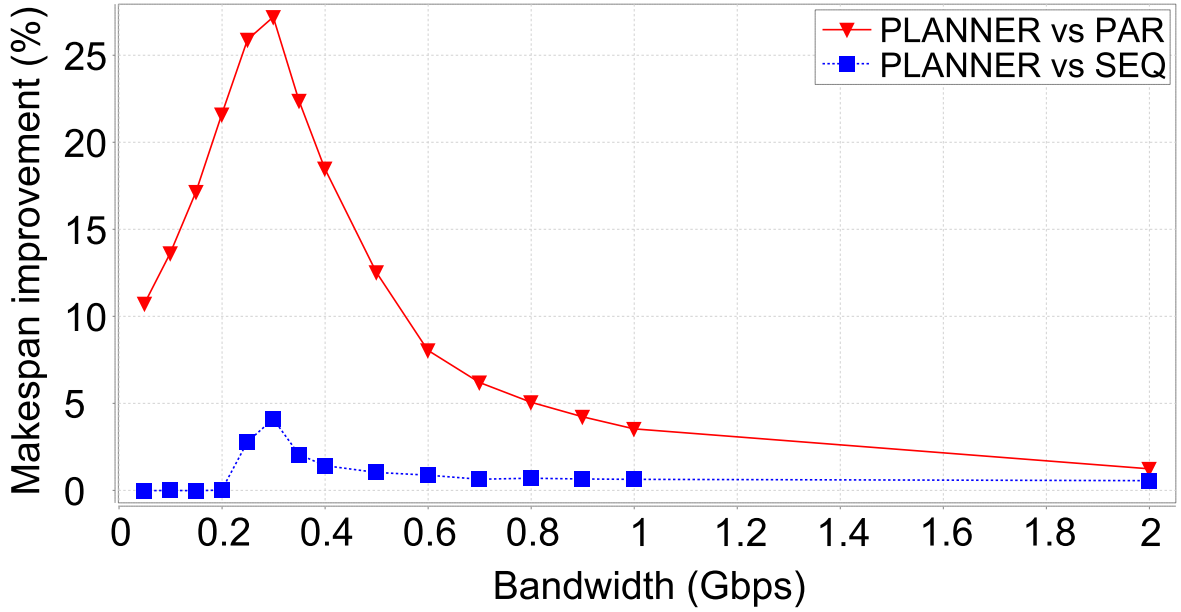
An input file is transferred entirely before the processing starts
An output file is transferred after the job is completely finished

Scheduling approaches

- **SEQ**: "send a job to a free CPU" + sequential
- **PAR**: "send a job to a free CPU" + parallel
- **PLANNER**: our approach + sequential

$$Makespan\ improvement = \frac{makespan_1 - makespan_2}{makespan_1}$$

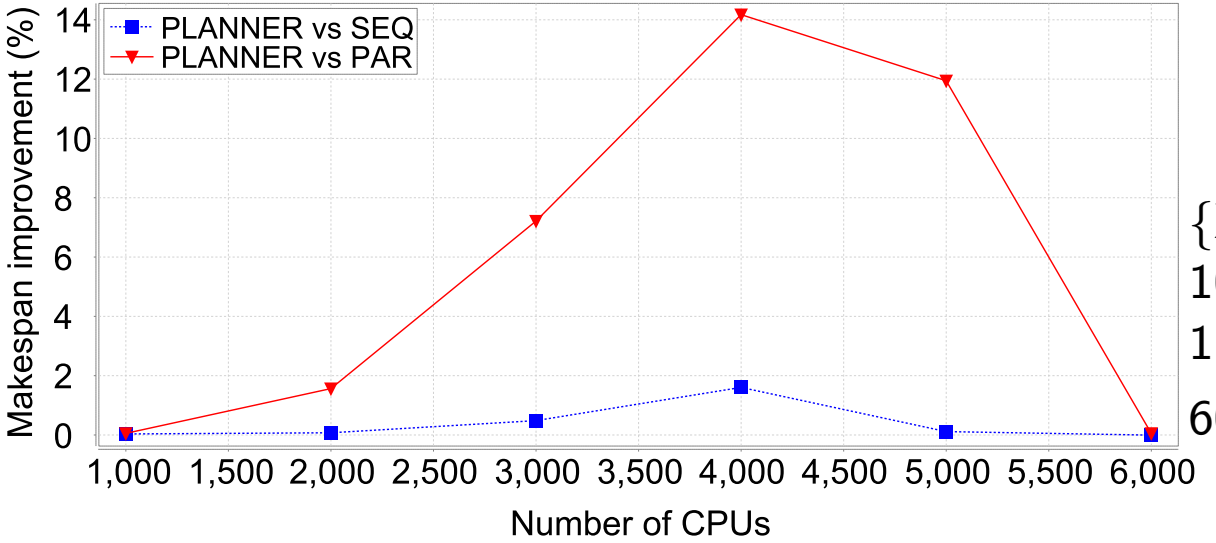
Single remote computational site (Makespan vs CPUs and Bandwidth)



{1,000 CPUs;
12 TB disk;
X Gbps }

7,000 jobs

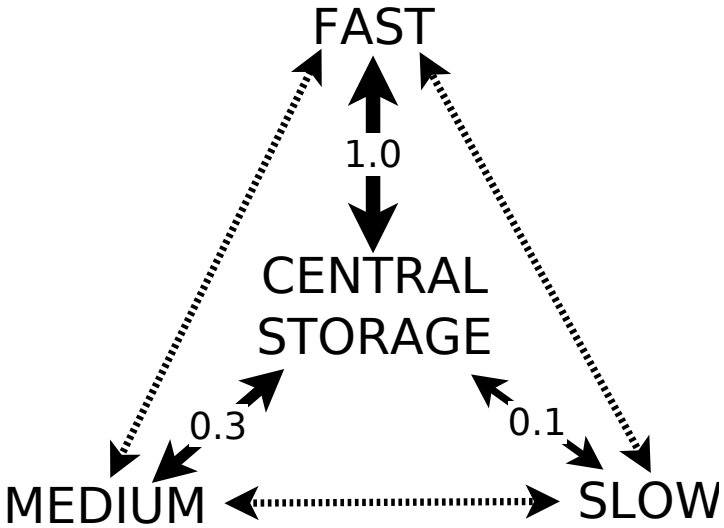
The planner broadens the scope of efficient Grid setup {CPUs, disk, bandwidth}



{X CPUs;
10 TB per 1,000 CPUs;
1 Gbps }

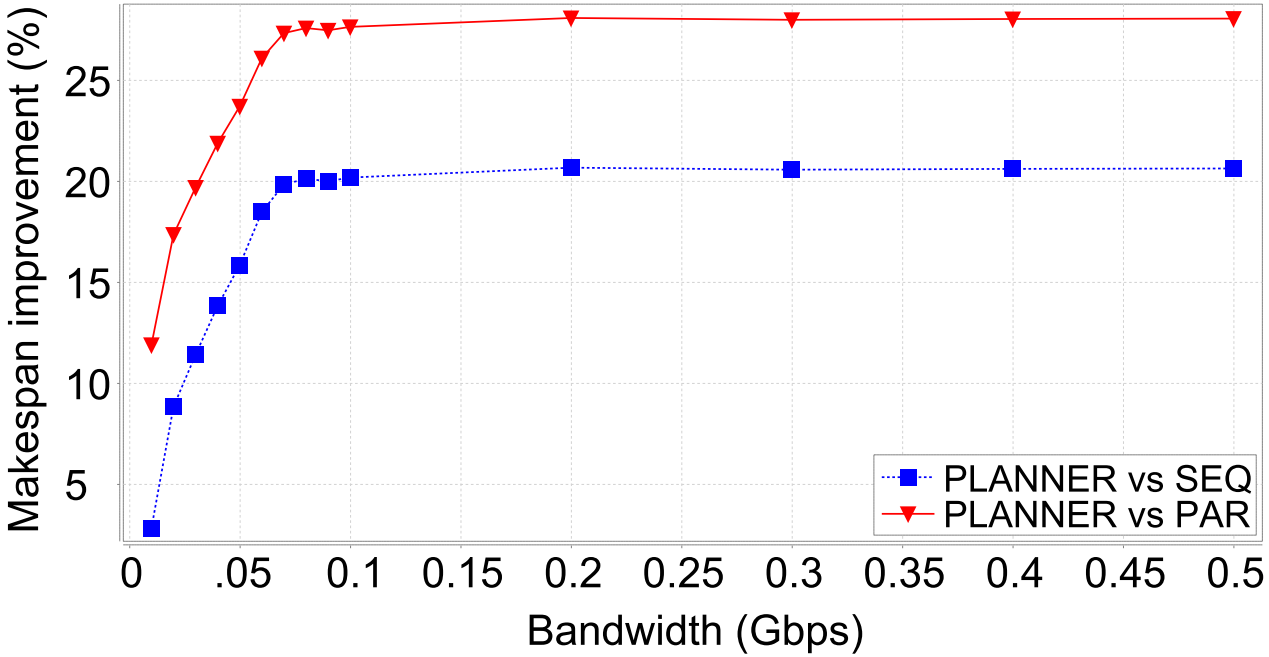
60,000 jobs

Multiple remote sites



- Each site {1,000 CPUs; 12 TB; ...}
- Changing bandwidth of dotted links
- 60,000 jobs

The PLANNER efficiently balances the network load



Simulations with background traffic

Network infrastructure is shared with other experiments (activities)

GridSim implementation

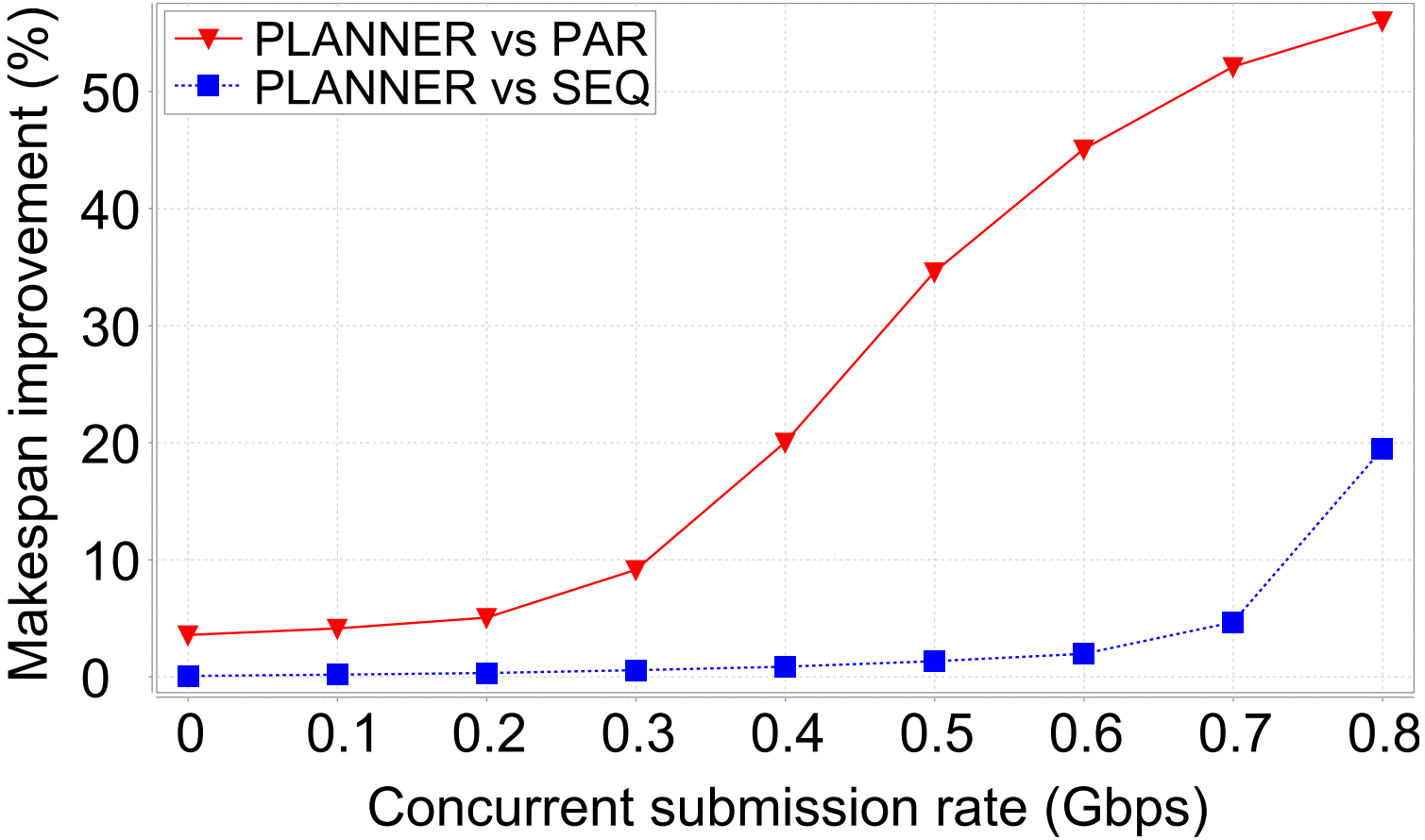
- Each site sends unrelated files to the other (selected) sites
- Number of unrelated files to be sent
- Time intervals between sending
- Size of unrelated files

Observations from the simulations

- Background traffic increases **transfer latency**
- Transferring files in advance becomes extremely efficient
- Larger ΔT becomes advantageous (6 – 12 h)

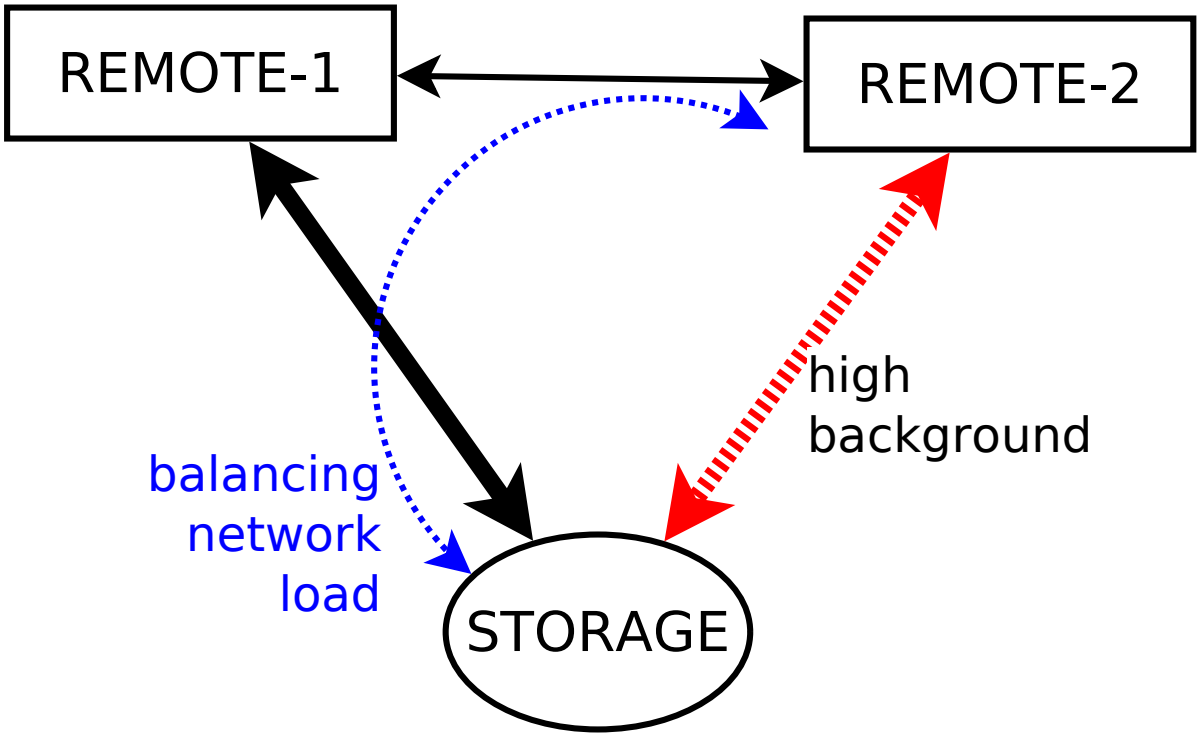
Single remote site with background traffic

{1,000 CPUs; 14 TB disk; 1 Gbps}; 7,000 jobs, shared link
Background: 12 Gb files every 1,000 seconds, changing number



The PLANNER utilizes a shared link more efficiently

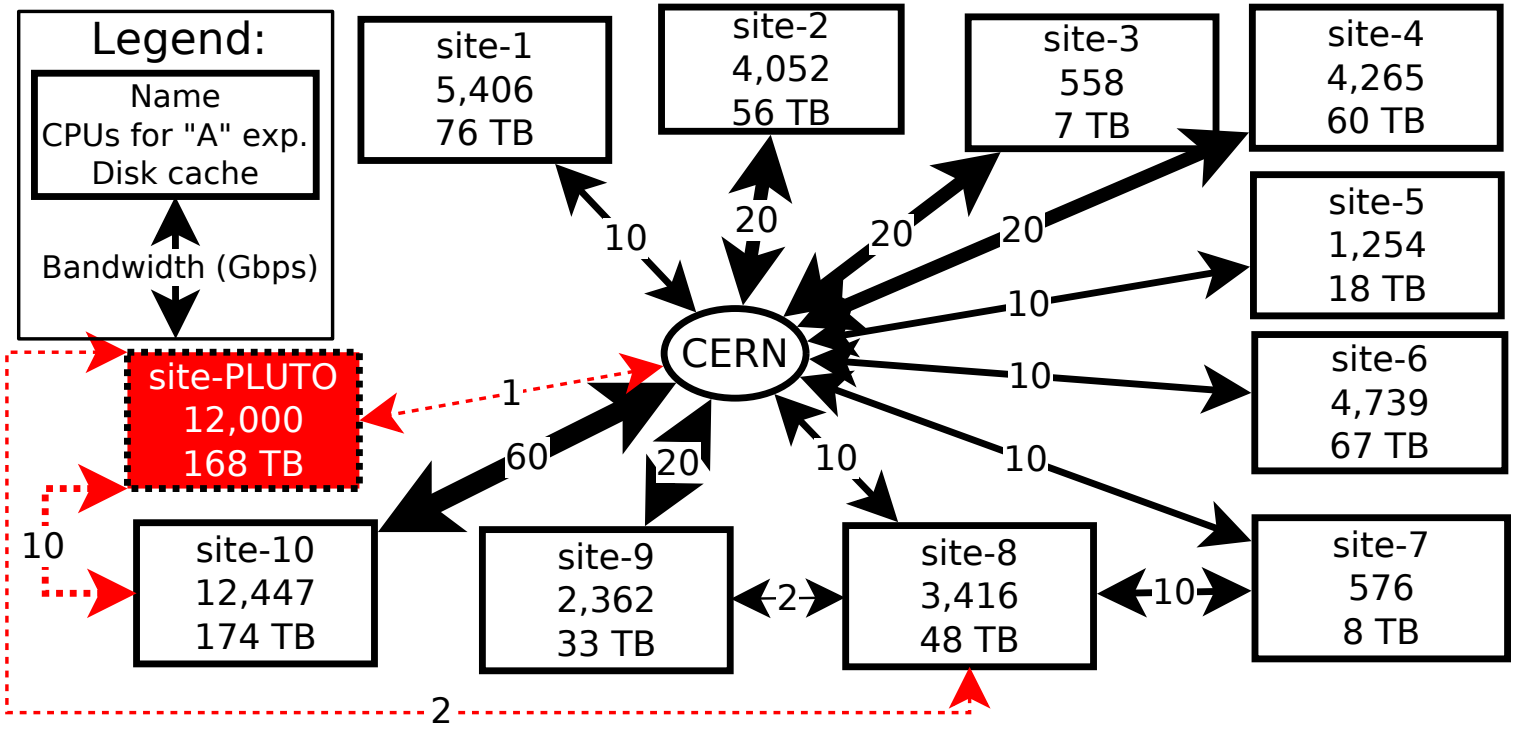
What about multiple remote sites & background traffic? (planned improvement)



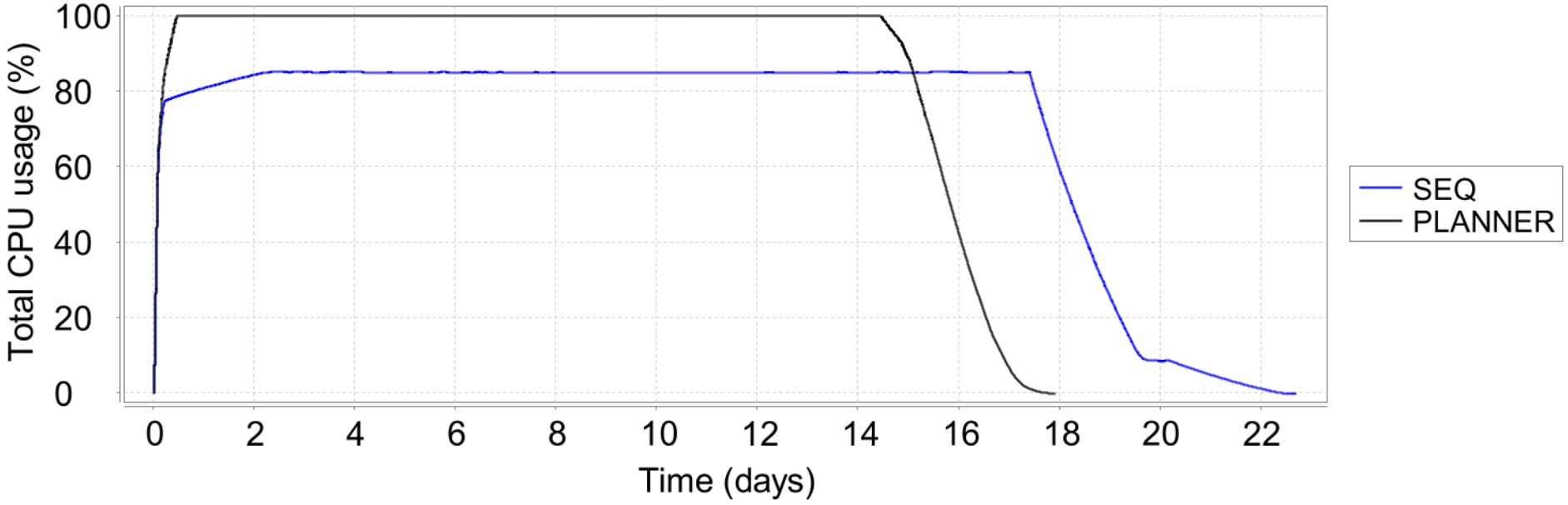
Testing scalability

- Grid of a larger experiment "A" Tier-1s (Data from online monitoring tools from CERN, November 2015)
- 558 – 12,447 CPUs at 10 sites (39,075 total), bandwidth 2 – 60 Gbps
- Consider a fraction of available disk space (cache) ∼ CPU number
- The designed parameters are well balanced

What if we add a site with poor connection to Tier-0 but large CPU pool?



Large scale simulations results (11 sites; 51,075 CPUs; 500 k jobs)



☹ Missing PAR: simulation too computationally demanding

- The **PLANNER utilizes more CPUs at the "PLUTO" site** by using indirect transfer paths (automatically). Makespan improvement = 21 %
- **Plan for (every) 12 h of data production is created in 7 ms**

Conclusion

Previous work [Makatun et al., 2015a]

- New job scheduling approach for data production has been presented
- GridSim simulations and data from the real infrastructure was used to validate the model
- Its implementation: a planner, handlers - validated against other classic scheduling approaches

Our proposed approach

- Systematically provides better makespan than classic job scheduling
- Can optimize makespans over a wide range of infrastructure configurations in {CPU, disk, network bandwidth}
- Model can deal with loaded (shared) networks and adapts
- Optimization and path not always intuitive - optimization on secondary network path automated, allowing to utilize sites with poor connectivity to central Tier-0
- Scalable (polynomial complexity on Grid size). Plan for 12 h of data production is created in 7 ms

Future Plans

- Simulations with data from more HENP experiments
- Study influence of site parameters: Find the region of optimal values in {CPU, disk, network bandwidth} phase-space and how our approach can broaden it
- Study the influence of internal planner parameters (such as ΔT). Ideally, make the planner to calculate those parameters for itself, using data of previously finished jobs
- Simulate more realistic cases for comparison (continuous I/O during job execution, mixing Tier-1 and Tier-2 sites, sites with storage only, underlying network structure, etc)
- Integration into existing distributed RMS
- Deployment to the data production system of the STAR experiment, testing, statistics collection

References

Buyya, R. and Murshed, M. (2002).
GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing.
The Journal of Concurrency and Computation: Practice and Experience (CCPE), 14.

Hajdu, L., Lauret, J., Didenko, L., Amol, J., Betts, W., Jang, H. J., and Noh, S. Y. (2016).
Automated finite state workflow for distributed data production.
In *ACAT2016*.

Makatun, D., Lauret, J., Rudová, H., and Šumbera, M. (2015a).
Distributed data production planning for High Energy and Nuclear Physics.
Journal of Scheduling.
(Submitted).

Makatun, D., Lauret, J., Rudová, H., and Šumbera, M. (2015b).
Model for planning of distributed data production.
In *Proceedings of the 7th Multidisciplinary International Scheduling Conference (MISTA)*, pages 699–703.

The end

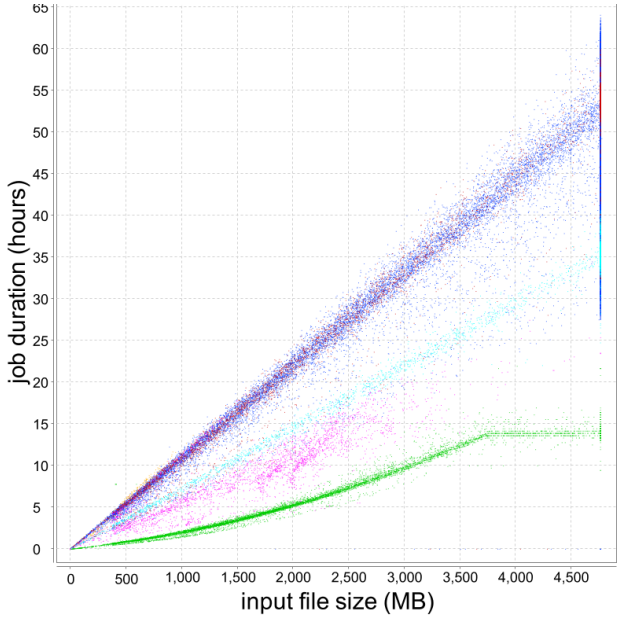
Jobs from monitoring database by filetype (filtered)

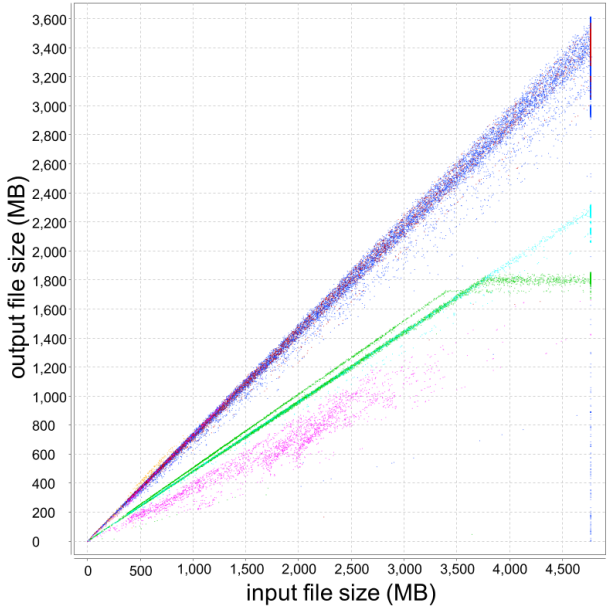
- ~77k jobs

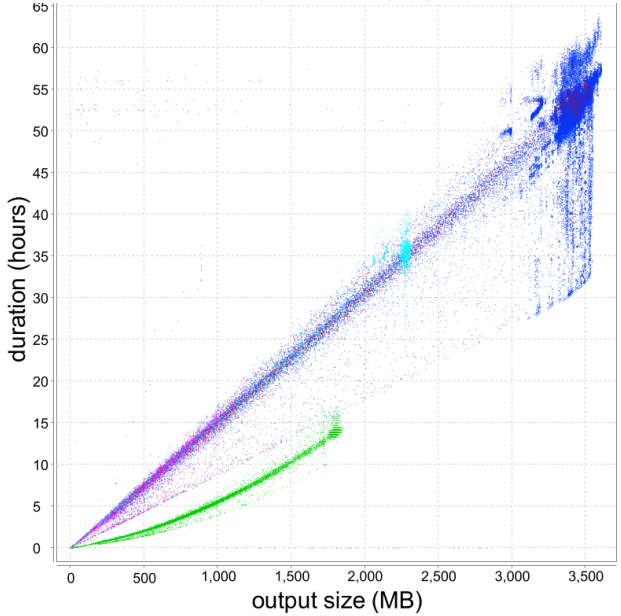
$$\alpha = \frac{jobDuration}{InputSize}$$

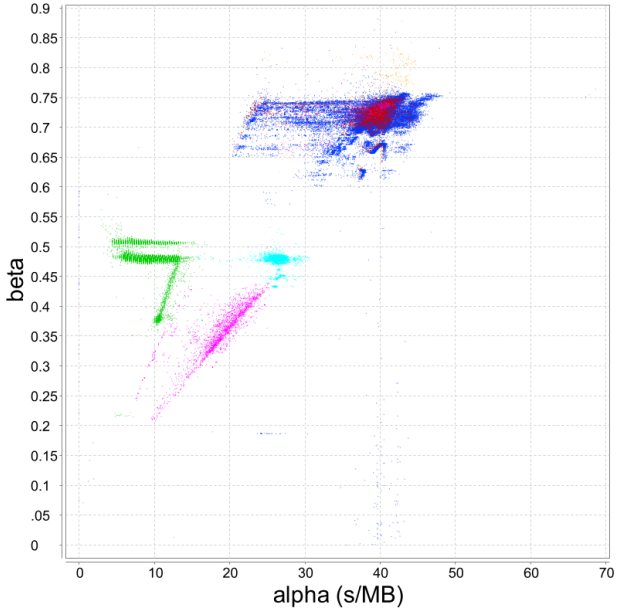
$$\beta = \frac{OutputSize}{InputSize}$$

- st_physics_
- st_laser_
- st_zerobias_
- st_fms_
- st_mtd_
- st_jet_
- st_fgt_
- st_daqtenk_
- st_tof
- st_hlt_



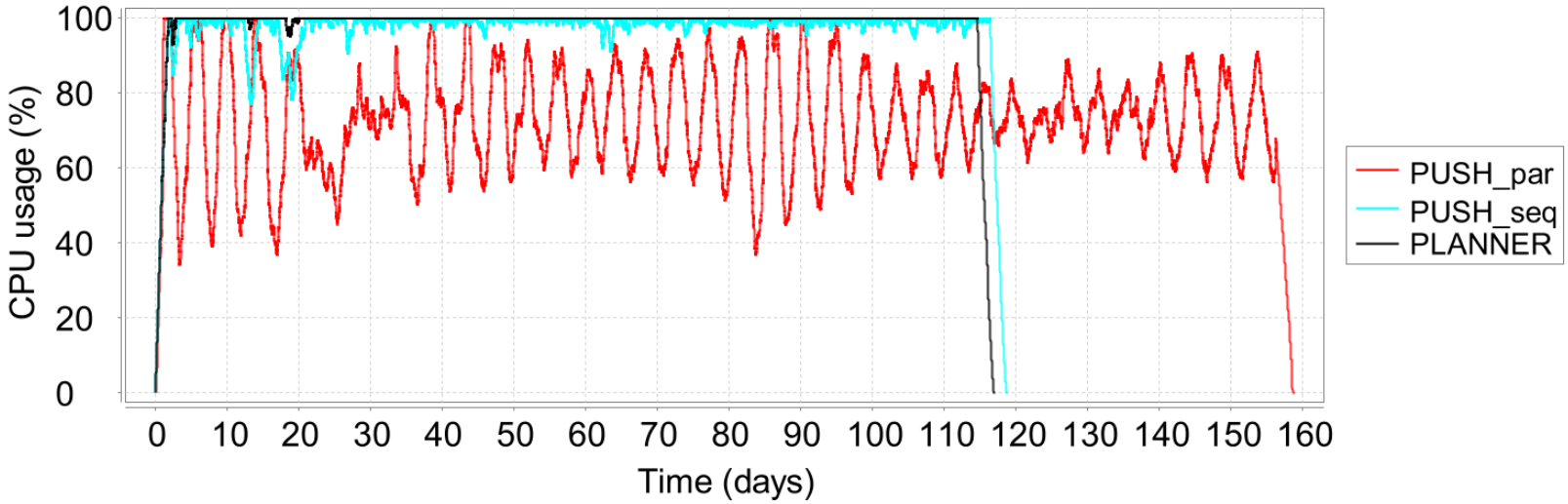






Single remote computational site (low bandwidth case)

60,000 jobs, 1,000 CPUs, 10 TB local disk, 250 Mbps connection to central storage



Disk and CPU usage by the PLANNER: