Distributed Data Processing for High Energy Physics

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

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

[2]Brookhaven National Laboratory, USA

[3]Czech Technical University in Prague, Czech Republic

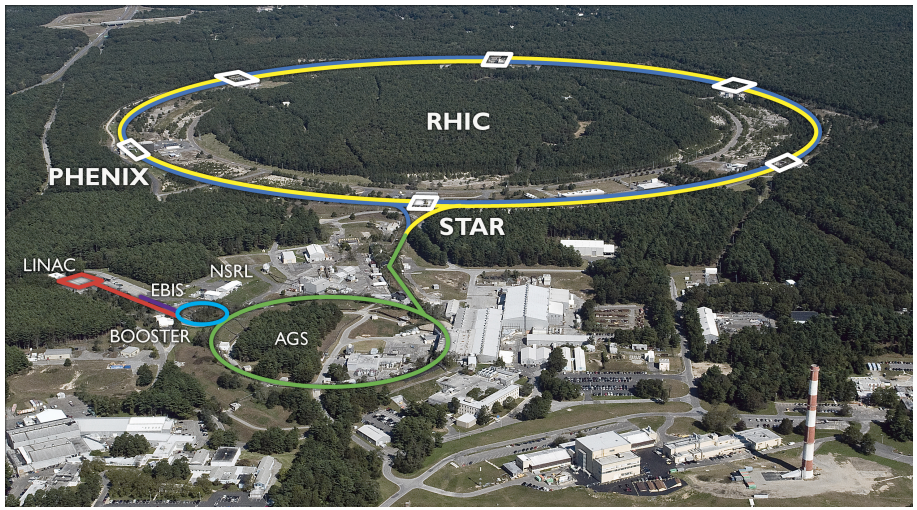
[4]Faculty of Informatics, Masaryk University, Czech Republic

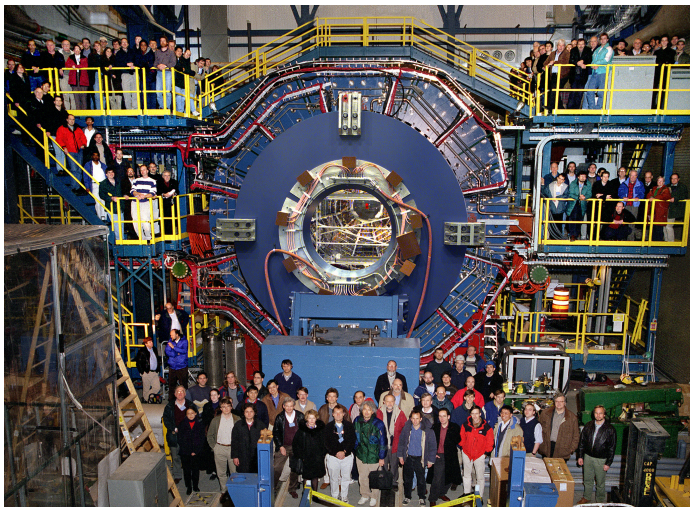
d.i.makatun@gmail.com

June 24, 2015

Outline

Computations in HEP: what do we compute?

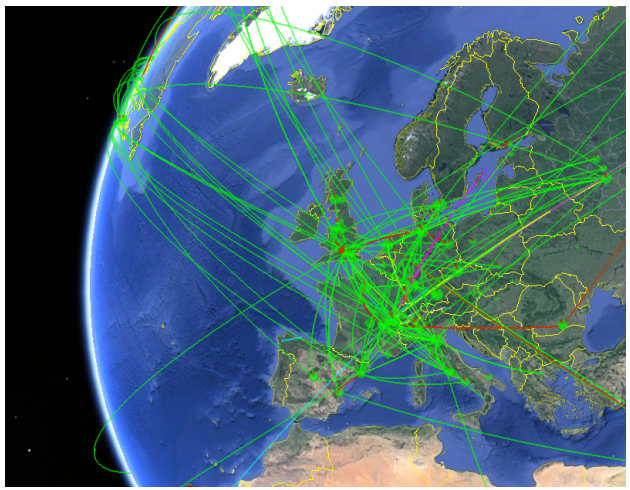


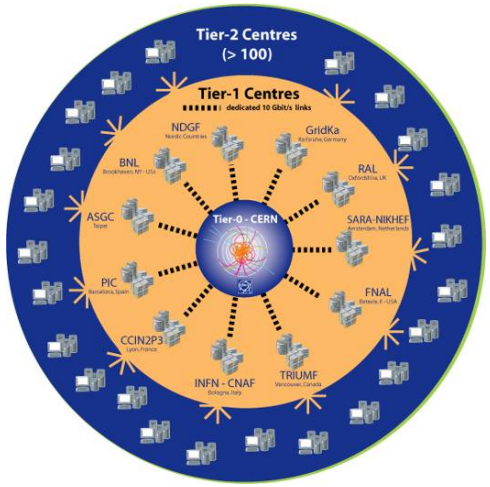


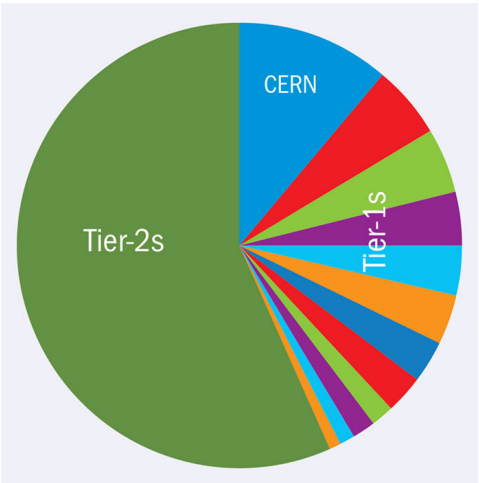
- Brookhaven National Laboratory (**BNL**) Long Island, NY, USA
- Relativistic Heavy Ion Collider (**RHIC**). In Gold-Gold ion collisions a quark-gluon plasma is created to study the primordial form of matter that existed in the universe shortly after the Big Bang.
- Solenoid Tracker at RHIC (**STAR**). Collisions occur millions of times per second. Events of size 200 MB are processed at input rates up to 100Hz. Output data rate is $\sim$ 30 MB/sec.

Computations in HEP: how do we compute?

Data Production: The raw output data is processed to reconstruct events ($\sim$ ones).
User Analysis: Then the reconstructed events are analyzed by scientists to discover new physics (each file many times).
All the data: raw, reconstructed and analysis output are stored.
31 PB of data stored on tape, $\sim$ 12 000 jobs running simultaneously (at RCF only).

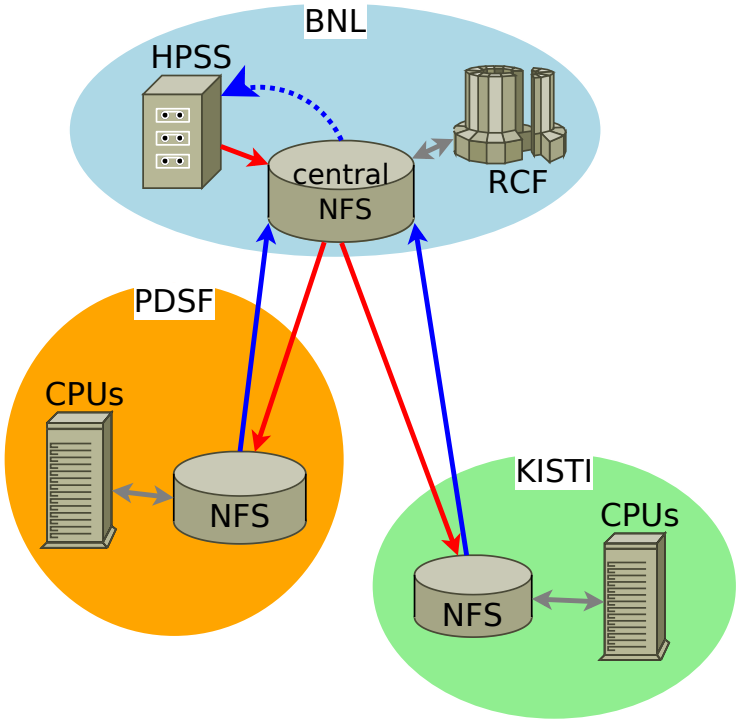






Case 1: Data production. Planning remote site usage

- RAW data is located at BNL.
- Computational resources are available at BNL and several remote sites.
- 1 job per file.
- 1 CPU per job.
- Input size $\approx$ Output size.
- Output file has to be transferred back to BNL.
- **How should we distribute a given set of files between sites to complete the processing faster?**



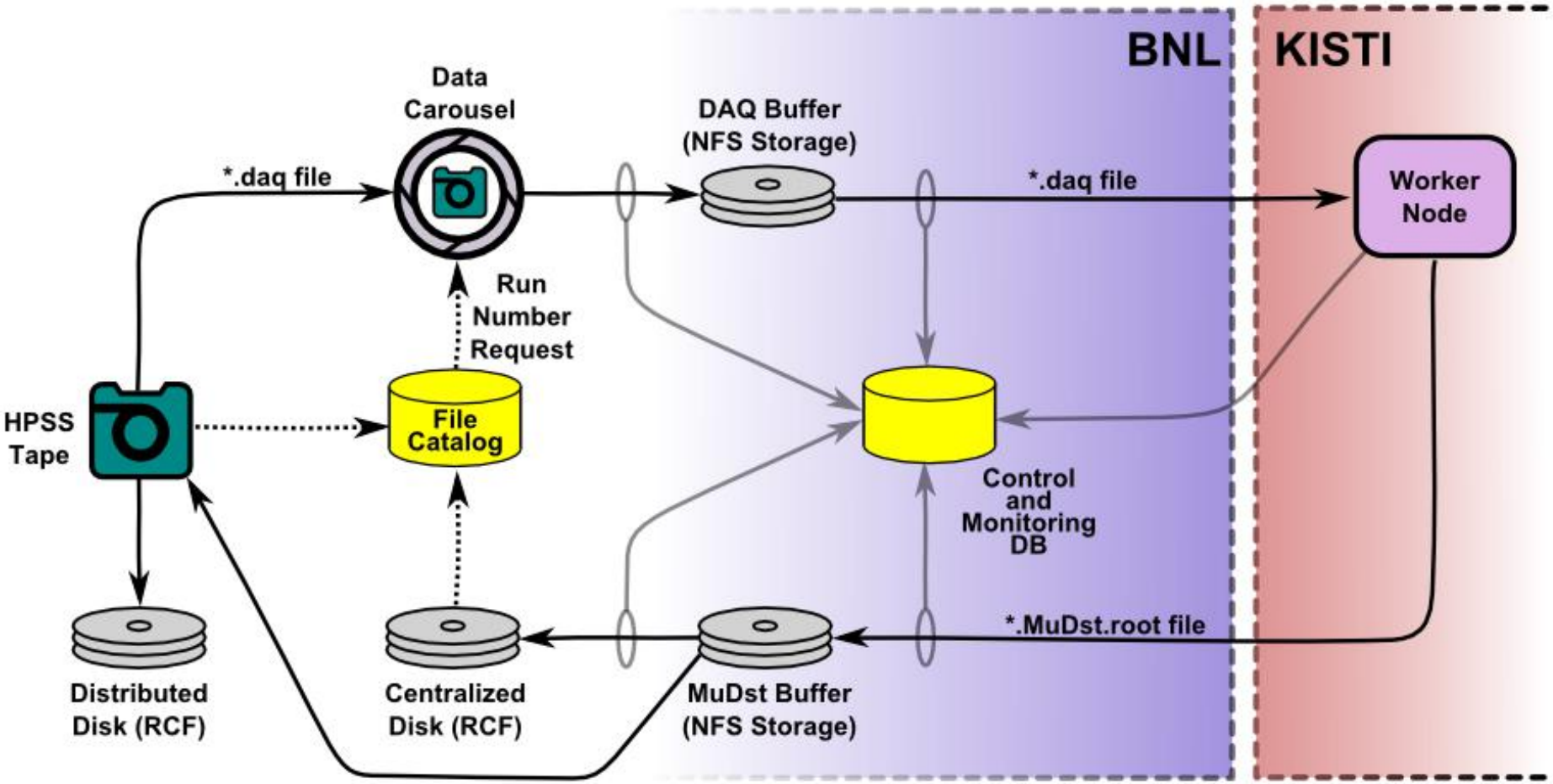
Manually adjust the number of remote jobs to meet the network throughput, **but** what if:

- More sites
- Changing network load

This should be automated.

Case 1: Example

STAR: setup for data production at a remote site (KISTI)



- For better efficiency an ad-hock setup is used. [*Korea Institute of Science and Technology Information KISTI*]

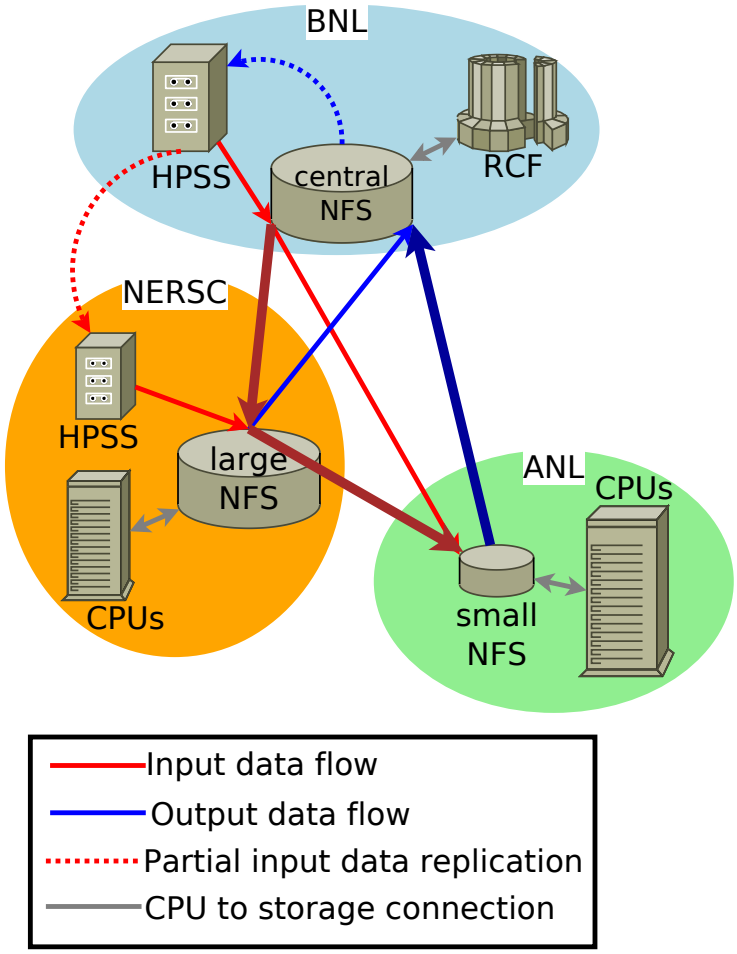
Case 2: Data production. Optimization

New dimensions of the problem

- Several possible data sources.
- Real network topology: shared links, **links between remote sites**.
- Limited storage at sites.
- **Which file source to select?**
- **What is the optimal transfer path?**

Example: data-production at ANL [Balewski et al. 2012]

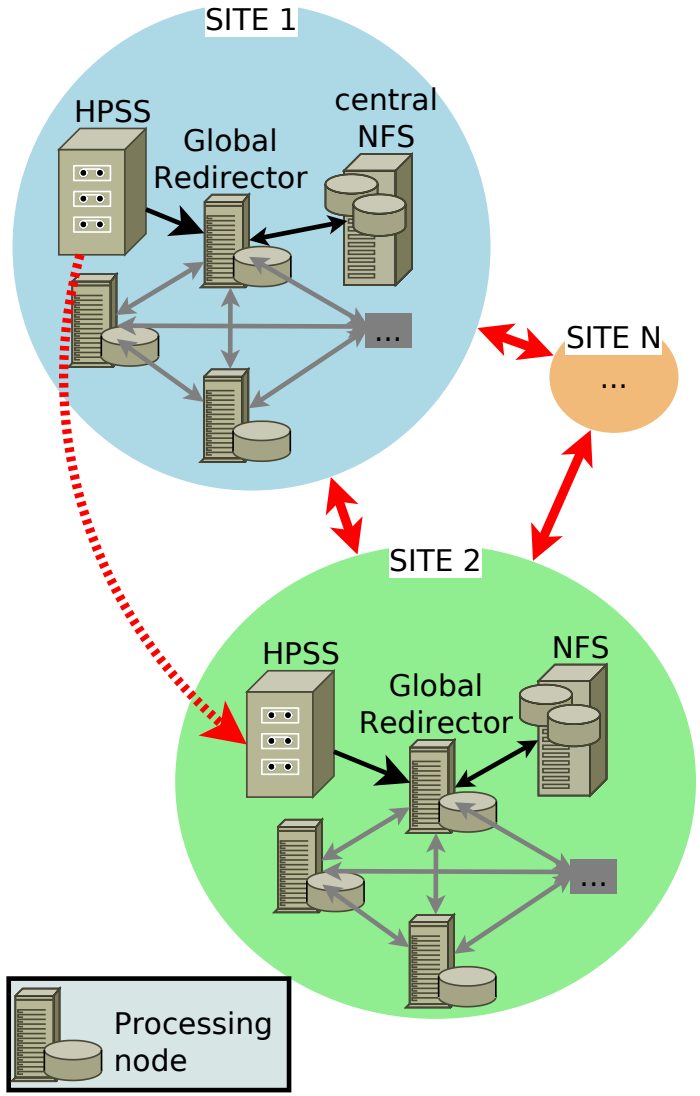
- ANL: many CPU's, but slow connection and small disk space.
- NERSC: fast connection, large disk.
- Optimization: Feed ANL from both BNL and NERCS sites.



Case 3: User analysis

New dimensions of the problem

- Many copies of files exist.
- Each file can be requested by multiple jobs.
- The size of output of analysis is negligible compared to input size.
- The processing time estimates are imprecise.
- **When and where to replicate the data?**



Motivation

[Zerola et al. 2012]

Efficient and controlled movement of replicated datasets within Grid to satisfy multiple requests in the shortest time.

- Select between several data sources.
- Create optimal transfer paths, merge shared transfer paths of the same file.
- Schedule transfers on links.

It was shown that global planning of **data-transferring** over Grid can outperform well known heuristics (e.g. P2P, Xrootd reasoning).

New goal: Problem extension

Global planning for **jobs coupled with transfers** in distributed environment.

Example of decisions

? Send a job to a site with slow connection **or** wait for a free slot at local site?

? Access data remotely **or** transfer it before the job starts?

Heuristics in use `[Pull a job when CPU slot is free]` will not give the answer.

Existing solutions (used in HENP)

Batch System + Distributed Data Management System (Independent)

- PBS, Condor. [Pull a job from the global queue]
- Xrootd, DPM. [The site which replies the first is selected as a source]

Data Trains (For user analysis)

- Group jobs by input data $\longrightarrow$ preplace data $\longrightarrow$ start jobs simultaneously $\longrightarrow$ kill latest x% of jobs.
- Train runs periodically ($\sim$ ones per day).
- Controlled by train operators.

Globus (Decoupling jobs and transfers)

- Sends jobs to data.
- Replicate most "popular" files. Relies on usage history.
- Where to replicate? When to replicate? No transfer planning.

Related work (not in use in HENP)

Bandwidth-Centric Allocation of Independent Tasks on Heterogeneous Platforms [Beaumont et al. 2001]

- Exact solution for maximum steady-state throughput.
- Grid network is modeled as a tree (no alternative paths). Single source/destination. Input path = Output path. Equal size of jobs/files.

XSufferage [Casanova et al. 2000]

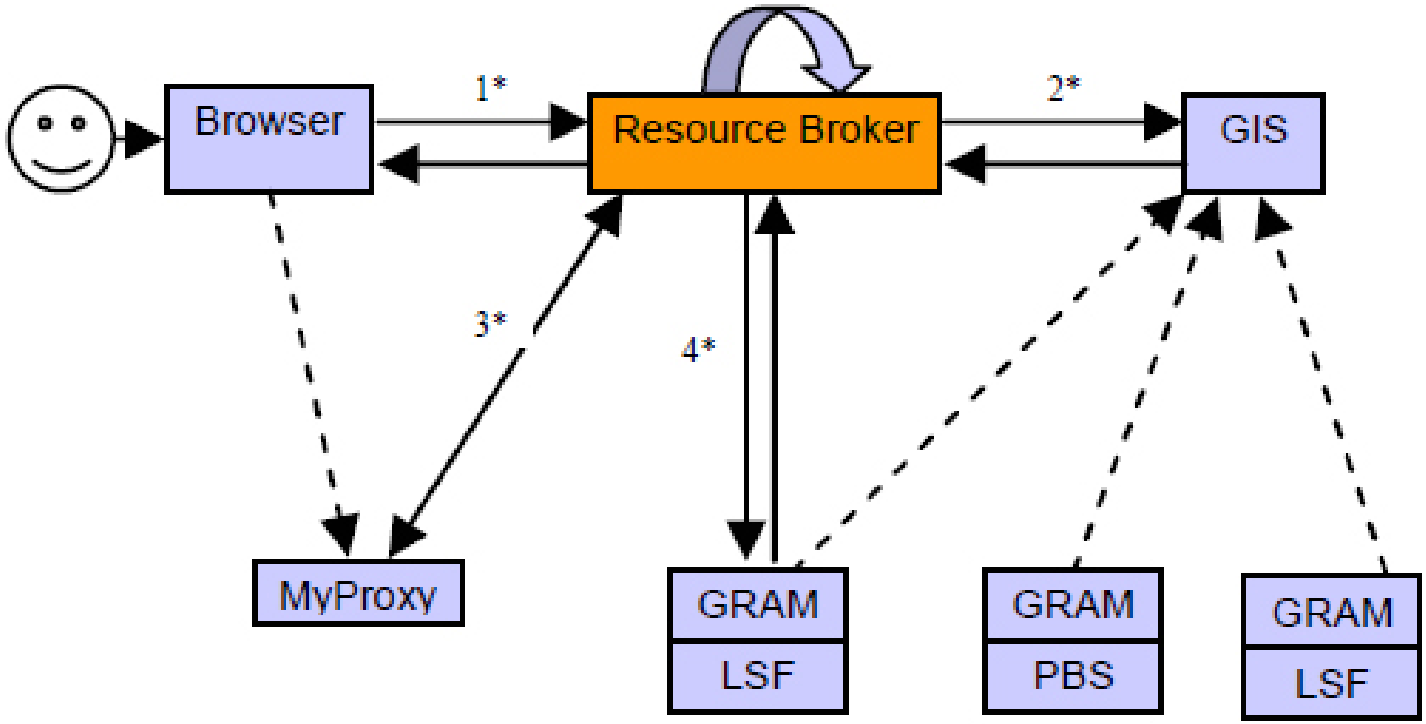
- Considers I/O transfer latency. Assigns jobs to hosts based on $Sufferage = SecondBestEstimatedMakespan - BestEstimatedMakespan$
- No path/source selection or transfer planning. Simplified network model. No storage model.

Storage Affinity [Santos-Neto et al. 2005]

- XSufferage + job replication: Executes copies of the same job at several clusters concurrently.
- Simplified network model. CPU waste $\sim 25 - 60\%$.

Distributed resource management system (RMS): Architecture (Globus example)

- The resource broker is acting as a middle tier between a user and the resources by doing resource matching and job submission for the user.



[Ranganathan and Foster 2002]

Data production problem

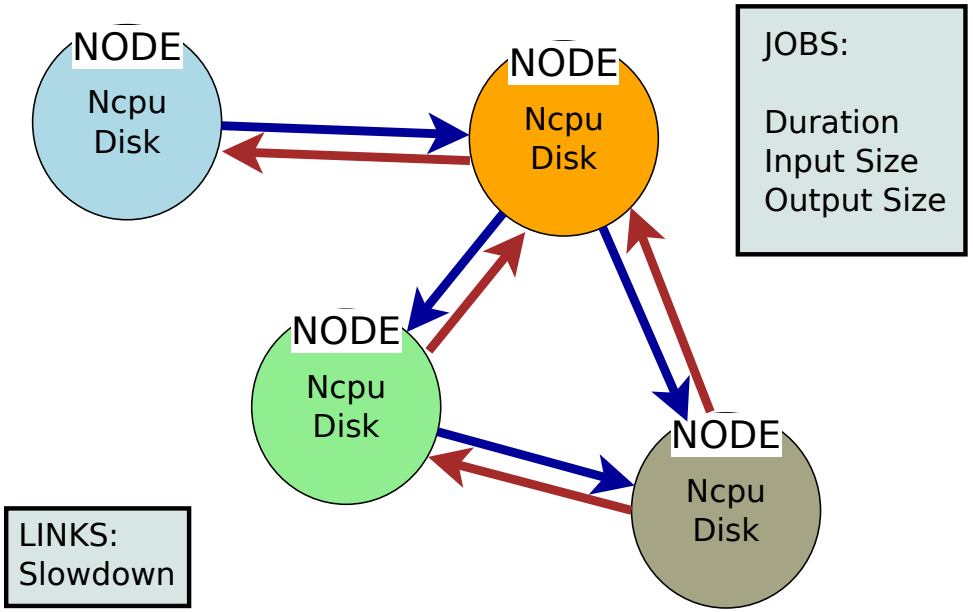
Create a global scheduler for Grid which will reason about:
1.data transferring, 2. CPU allocation, 3. data storage.

Optimization

- None of the resources (network links, data storages and CPUs) are over-saturated at any moment of time.
- The jobs are executed where the data is pre-placed.
- No excessive transfers or data replication.
- Minimal overall makespan for a given set of tasks.

To solve the problem we applied Constraint programming due to its techniques for scheduling, planning and optimization.

Data-production problem: Input



Assumptions

In previous work [Zerola et al. 2012] it was proved that:

- There is advantage to plan and schedule jobs by chunks (split the whole set by portions).
  - \+ More adaptability to changing environment.
  - \+ Faster plan creation.
- The network links can be considered as unary resources: one file-transfer at a time over link.

Solving procedure overview

1. **Initialization Stage**. Estimate *TimeLimit*.
2. **Planning Stage**. Instantiate a part of domain variables with the help of simplified constraints.
   a. Assign jobs to computational nodes.
   b. Select transfer paths for input and output files.
   c. Additional constraints: load balance, cycle avoidance, *etc*.
   d. Find a solution for the sub-problem.
3. **Scheduling stage**: define start time for each operation.
   a. Constraints on order of operations.
   b. Cumulative constraints.
   c. Minimize target function: (e.g. makespan).

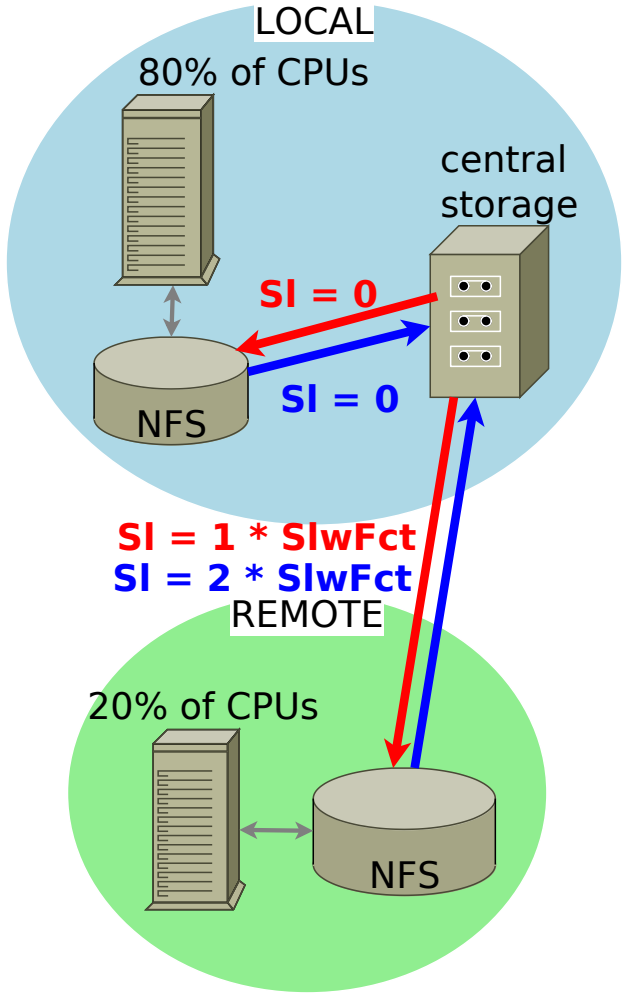
Simulations based on log data: problem setup

Input for simulations

- Parameters of 2000 jobs taken from log files of data production for STAR at KISTI.
- Jobs scheduled by chunks of 200.
- Slowdown of the link to the remote site proportional to the **slowdown factor**.
- Makespan compared to local processing.

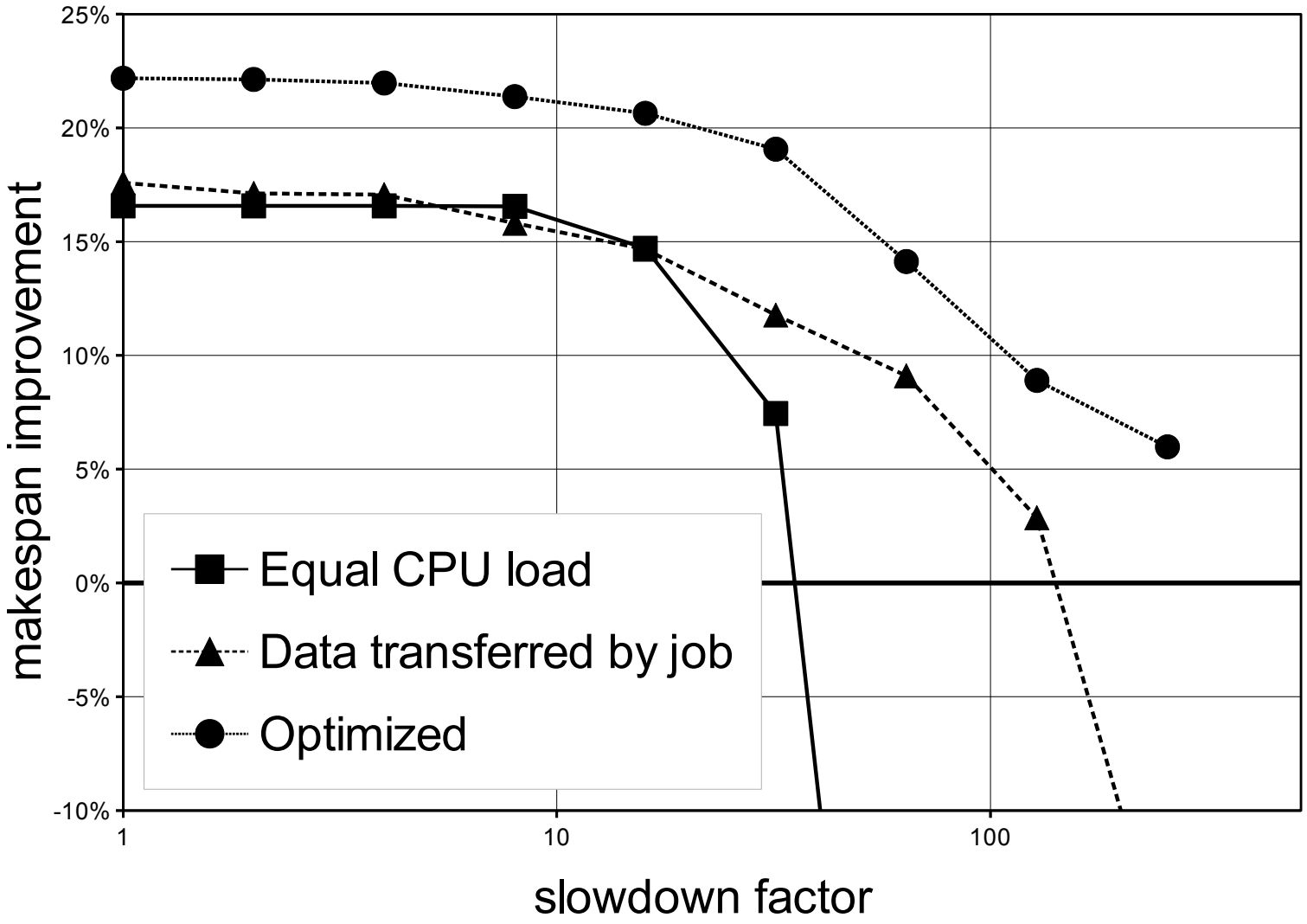
Tested algorithms

- Equal CPU load.
- Data transferred by job.
- Optimized.



Constraints for storage capacity are omitted.

Simulations based on log data: results



Results of simulations

- In simulated environment, where a remote site has the same CPU number as a local site, but data transfer overhead is comparable to job duration:
  - Scheduling with consideration of transfer overhead can reduce makespan by 15%.
- The **simulations based on log files:**
  - systematically provides a **smaller makespan**
  - **adapts** to the increase of transfer overheads better then the other simulated heuristics.
- Proposed approach can provide **automation, optimization** and **adaptation** to fluctuating resources.

Motivation

Drawbacks of CP model

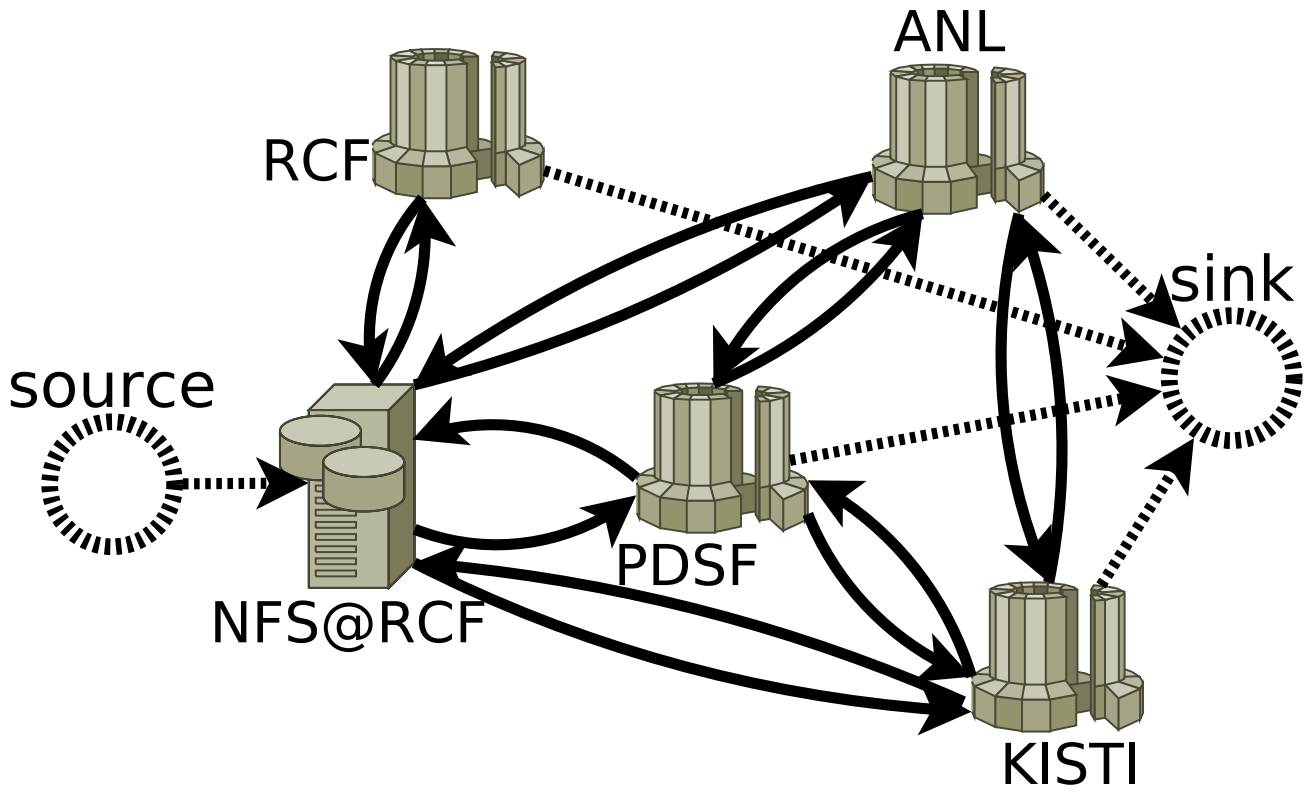
- Slow performance (NP-hard problem).
- Unnecessary calculations:
  - Path for each job (repeatedly).
  - Node selection for a particular job.
  - Order of jobs.

Network flow model

- Idea: plan resource load only and then distribute particular jobs accordingly. Planning time interval ΔT, *Flow* - amount of data, link capacity - maximum bandwidth.
- Network flow maximization problem can be solved within polynomial time.

Input transfer planning: How much data can be transferred during the next planning interval ΔT?

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



Output flow problem can be formulated similarly.

Solving procedure

Problems for input and output transfers can be solved independently under assumptions:

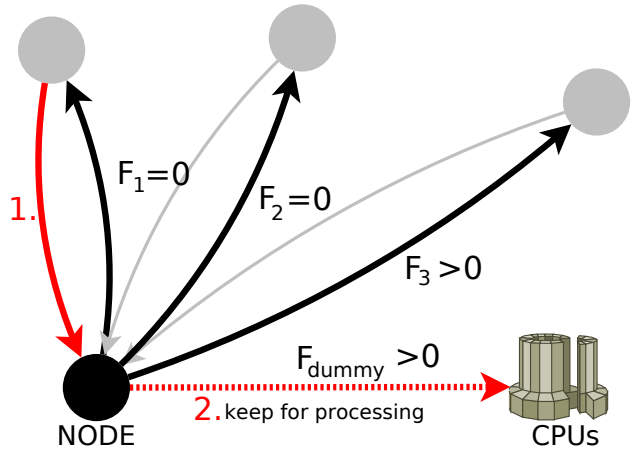
- Full-duplex links,
- In a steady state at each node
  $Processed\_input\_data = \beta \cdot Created\_output\_data$,
  where $\beta = const \leq 1$.

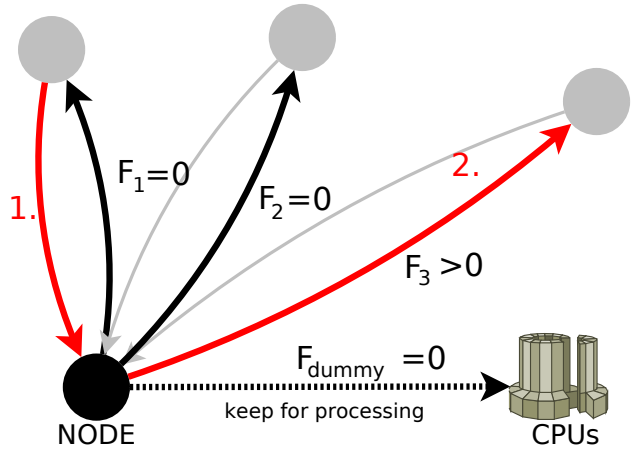
Solving procedure

1. Calculate capacities of (dummy) edges using monitoring data.
2. Solve the problem for output data flows.
3. Recalculate remaining network capacity.
4. Solve the problem for input transfers.

Plan execution

- A local handler at each node receives the plan:
  - Flows of outgoing edges - how much data of each type should be send over that link.
  - Flow of dummy edges - how much data should be processed at this node.
- When a new file arrives, the handler decides according to the plan and current state:
  - To process the file (to transfer over the dummy edge)
  - OR to forward it over one of the links.
- Decrease remaining flow of the link by the size of the file after transfer.





Network flow model: summary

- Feasible data distribution such that CPUs are busy with jobs while not exceeding disk capacities.
- Scalability and adaptability to changing environment.
- A planer using the model has been implemented and tested.
- Simulations using real log data of the STAR experiment are being implemented using GridSim (modeling tool widely used in Grid community).
- Extend the model to meet the broader scope of use cases: several data sources, real network topology, heterogeneous resources and load.
- Test the planner performance in order to enable online scheduling.
- Deploy to data production system of the STAR experiment. Test and collect statistics.

Conclusion

- Automated and scalable **planning and optimization of distributed computations** are highly required in data intensive computational fields such as High Energy and Nuclear Physics. Resent works has revealed the potential of **global planning** for this task.
- A **constraint programming model** for scheduling of data production in Grid was formulated.
- The **simulations based on log data** has shown that the proposed approach systematically provides a smaller makespan and adapts to the increase of transfer overheads better then the other simulated heuristics.
- **Network flow model** has been developed to address the limitations of CP model. Its further development and testing is ongoing.
- Performance of a wide scope of **caching algorithms** was simulated using access patterns derived from log files of HENP experiments. The most appropriate ones were selected for data management.

Aims of the PhD research

to develop and implement a general methodology for **planning of data intense computations in a distributed environment** with primary focus on **data production** in HENP.

Goals

- **Mathematical model.** General mathematical model for data production planning. Suitable level of abstraction, extensible.
- **Scalable planer:** Petabytes of data, hundreds of computational sites, real-time planning.
- **Adaptability to dynamic changes:** Background network traffic, cluster load by other tasks, downtime, errors, failures.
- **Extended versions of the problem.** Several data sources, real network topology, several experiments (users), user analysis case, resource heterogeneity.
- **Evaluation using real log data.** STAR data production, other HENP experiments, scalability tests.
- **Deployment.** Integration into distributed RMS. Deployment to the data production system of the STAR experiment. Testing, statistics collection.

The end of the presentation

Thank You for Your attention.

Publications

Publications

[1] Dzmitry Makatun et al. “Model for planning of distributed data production”. In: *Proceedings of the 7th Multidisciplinary International Scheduling Conference (MISTA 2015)*. Accepted. 2015.

[2] Dzmitry Makatun et al. “Planning for distributed workflows: constraint-based coscheduling of computational jobs and data placement in distributed environments”. In: vol. 608. 1. 2015, p. 012028.

[3] Dzmitry Makatun, Jérôme Lauret, and Michal Šumbera. “Study of cache performance in distributed environment for data processing”. In: *Journal of Physics: Conference Series*. Vol. 523. IOP Publishing. 2014, p. 012016.

[4] Dzmitry Makatun. “Distributed Data Processing: Coscheduling of Jobs and Data Placement”. In: *Doktorandské dny 2014. Sborník workshopu doktorandů FJFI oboru Matematické inženýrství*. (České vysoké učení technické v Praze). Ed. by Pavel Ambrož and Zuzana Masáková. Praha: ČVUT-výroba, 2014, pp. 115–125. ISBN: 978-80-01-05605-9.

[5] Dzmitry Makatun. “Distributed Data Processing in High-Energy Physics”. In: *Doktorandské dny 2013. Sborník workshopu doktorandů FJFI oboru Matematické inženýrství*. (České vysoké učení technické v Praze). Ed. by Pavel Ambrož and Zuzana Masáková. Praha: ČVUT-výroba, 2013, pp. 151–161. ISBN: 978-80-01-05379-9.

[6] Dzmitry Makatun. “Distributed Data Processing in High-energy Physics”. In: *Doktorandské dny 2012. Sborník workshopu doktorandů FJFI oboru Matematické inženýrství*. (České vysoké učení technické v Praze). Ed. by Pavel Ambrož and Zuzana Masáková. Praha: ČVUT-výroba, 2012, pp. 155–167. ISBN: 978-80-01-05138-2.

Presentations

[7] Dzmitry Makatun. *Planning for distributed workflows: coscheduling of computational jobs and data placement in distributed environments*. STAR Regional Meeting. Prague, 2015.

[8] Dzmitry Makatun. *Distributed Data Processing for High Energy Physics*. Seminar at the Faculty of Informatics of Masaryk University. Brno, 2014.

[9] Dzmitry Makatun. *Distributed job scheduling*. The 2nd Optimization Summer School by National Information Communications Technology Research Centre of Australia (NICTA). Kioloa, 2014.

[10] Dzmitry Makatun. *Caching algorithms simulation*. Thematic CERN School of Computing. Split, Croatia, 2013.

[11] Dzmitry Makatun. *Cache performance in a distributed environment*. STAR Regional Meeting. Prague, 2013.

References

Korea Institute of Science and Technology Information KISTI. http://en.kisti.re.kr/.

Balewski, Jan et al. (2012). "Offloading peak processing to virtual farm by STAR experiment at RHIC". In: *J. Phys.: Conf. Ser.* 368.012011.

Zerola, M et al. (2012). "One click dataset transfer: toward efficient coupling of distributed storage resources and CPUs". In: *J. Phys.: Conf. Ser.* 368.012022.

Beaumont, Olivier et al. (2001). "Bandwidth-centric allocation of independent tasks on heterogeneous platforms". In: *Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002*. IEEE, 6–pp.

Casanova, Henri et al. (2000). "Heuristics for scheduling parameter sweep applications in grid environments". In: *Heterogeneous Computing Workshop, 2000.(HCW 2000) Proceedings. 9th*. IEEE, pp. 349–363.

Santos-Neto, Elizeu et al. (2005). "Exploiting replication and data reuse to efficiently schedule data-intensive applications on grids". In: *Job Scheduling Strategies for Parallel Processing*. Springer, pp. 210–232.

Ranganathan, K and I Foster (2002). "Decoupling computation and data scheduling in distributed data-intensive applications". In: *11th IEEE International Symposium on High Performance Distributed Computing*, pp. 352–358.

Troubil, Pavel and Hana Rudová (2011). "Integer linear programming models for media streams planning". In: *ICAOR* 11, pp. 509–522.

Enabling Caching

- Performance of cache algorithms implemented with watermarking concept was simulated for a wide scope of cache size and low marks. 3 access patterns of 2 different experiments were used as input for simulations. 27 algorithms were tested in 90 simulation setups.

Motivation

- Cache of data-transfer tools.
- Management of local data replicas.

Access patterns used for simulation

STAR1: RCF@BNL, **Tier-0** for STAR experiment, Xrootd log, user analysis, 3 months period (June-August 2012).

STAR2: RCF@BNL, **Tier-0** for STAR experiment, Xrootd log, user analysis, 7 months period (August 2012 - February 2013).

GOLIAS: FZU Prague, part of **Tier-2** of ATLAS. ATLAS and AUGER experiments, DPM log, user analysis + production, 3 months period (November 2012 - February 2013). AUGER makes less than 1% of total requests.

Selected caching algorithms

- **First-In-First-Out (FIFO)**: evicts files in the same order they entered the cache.
- **Least-Recently-Used (LRU)**: evicts the set of files which were not used for the longest period of time.
- **Least-Frequently-Used (LFU)**: evicts the set of files which were requested less times since they entered the cache.
- **Most Size (MS)**: evicts the set of files which have the largest size.
- **A**daptive **R**eplacement **C**ache (**ARC**). 2 lists: L1 - files with *access count* $= 1$, and L2 - files with *access count* > 1. LRU is applied to both list. The length of each list depends on $p = \textit{cache hits in } L1 / \textit{cache hits in } L2$.
- **L**east **V**alue based on **C**aching **T**ime (**LVCT**). Deletes files according to the value of the Utility Function.

$$UtilityFunction = \frac{1}{CachingTime \times FileSize} \tag{1}$$

where **Caching Time** of a file F is the sum of size of all files accessed after the last request for the file F.

What caching algorithm is the best?

Average improvement over FIFO

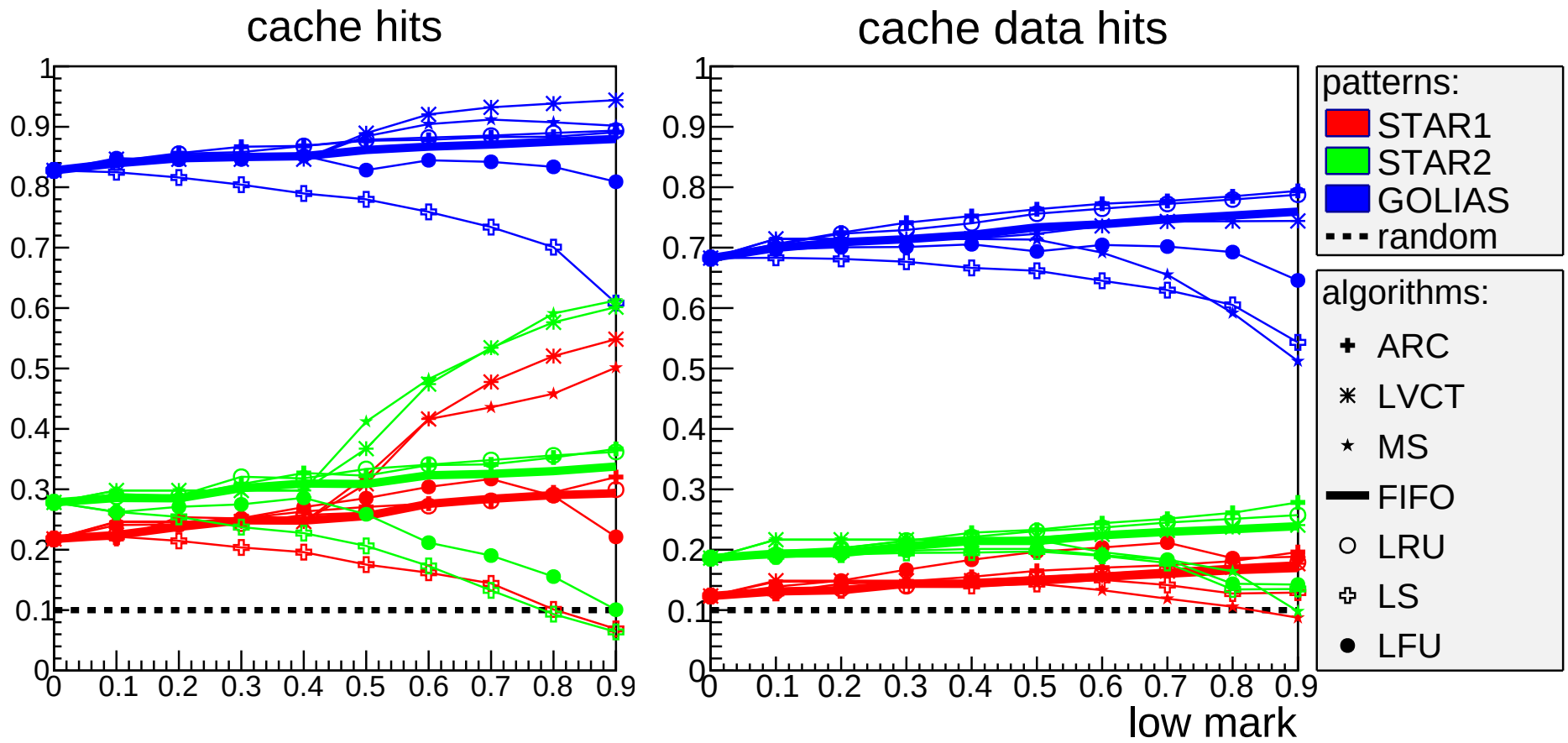
Algorithm	cache hits	cache data hits
MS	116 %	-20 %
LRU	8 %	5 %
LFU	-27 %	-18 %
ARC	13%	11%
LVCT	86 %	2 %

For studied access patterns

- Regardless of the cache size, Tier-level and specificity of experiment the LVCT and ARC appear to be the most efficient caching algorithms.
- If the goal is to minimize makespan due to a transfer startup overhead the LVCT algorithm should be selected.
- If the goal is to minimize the network load the ARC algorithm is an option.

Dependence of cache performance on low mark

cache size / storage size = 0.1 ,high mark = 0.95



- Difference between Tier-2 and Tier-0 leads to distinct cache performance.
- With higher low mark the number of clean-ups increases as well as overall performance.