Learning low-wastage memory allocations for IceProd tasks

Carl Witt <wittcarx@informatik.hu-berlin.de>
Jakob van Santen <jakob.van.santen@desy.de>
Ulf Leser <leser@informatik.hu-berlin.de>

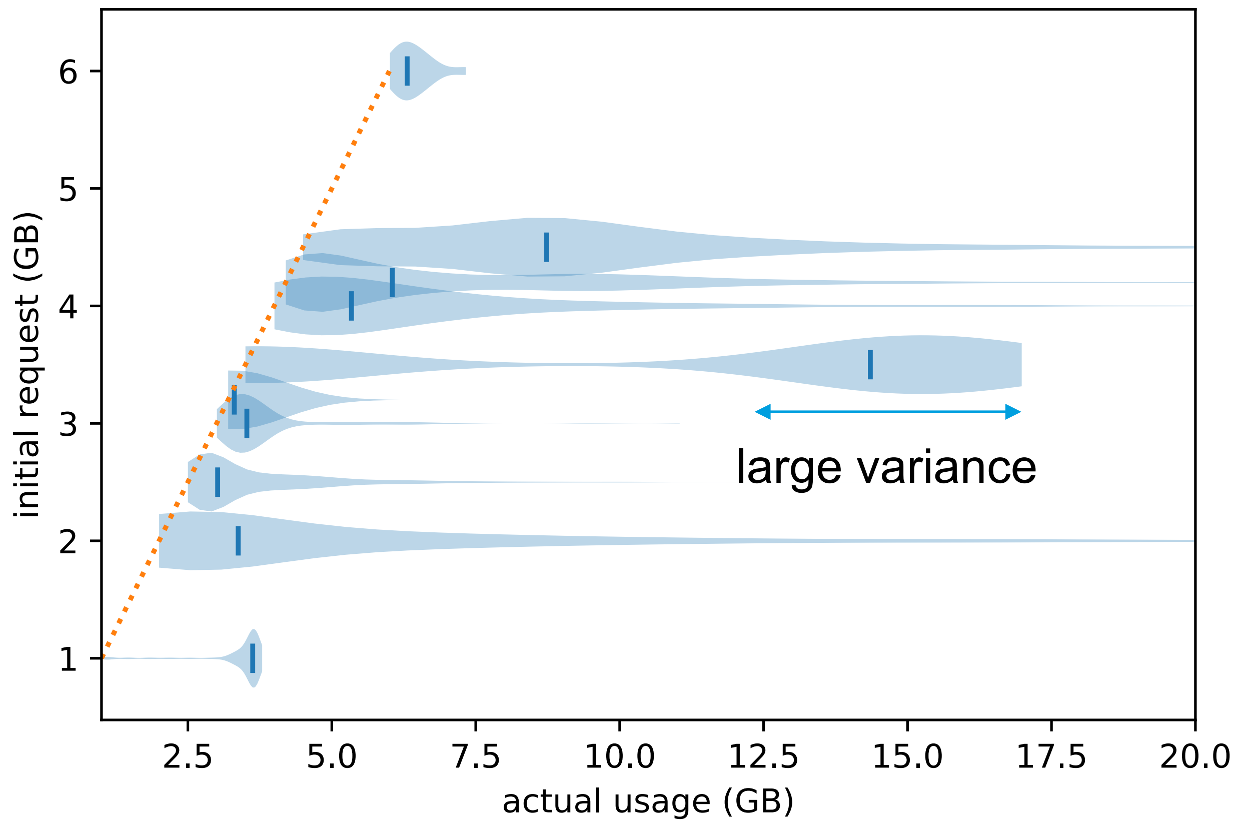
HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

Humboldt-Universität Informatik

DESY.

The Problem

- Initial task memory requirement is a guess. Tasks are killed if usage exceeds requests*.
- Actual memory requirement has to be discovered by trial and error.
- Without checkpointing, this wastes compute time and adds scheduling/startup overhead.



*Memory limits apply to an entire multi-core pilot; tasks can get away with overuse if other tasks on the same pilot under-use.

Why this happens

- Benchmarking a dataset configuration can take arbitrarily large amounts of human time
- Each concrete task has its own random number stream that may trigger memory allocations -> requirements unknown until all tasks are run
- High-memory tasks are somewhat rare, and retry strategy ensures that tasks eventually finish (except when they don’t)

Can we do better?

(i.e. at least as well as a novice human babysitting jobs full-time)

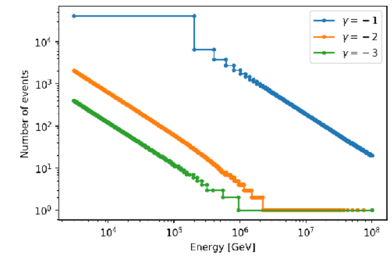
A previous attempt

- Beat down energy-dependent variance by partitioning power laws into narrow segments (similar to IceTop generation strategy)
- Good:
    - More predictable memory usage (energy range is a function of job index)
- Less good:
    - Relationship between energy and memory usage depends on dataset configuration -> still requires a human to benchmark
    - Failed tasks create noticeable gaps in the generated energy range*
    - Only used for a single Gen2 dataset

Partitioning a power law

- Instead of simulating the same spectrum N times (files), divide spectrum into N energy ranges
- Much smaller variance in input energy
- [More] predictable memory requirements
- Transparent to the end user: sum of all files is a simple power law
- But: need to be careful to use the entire dataset

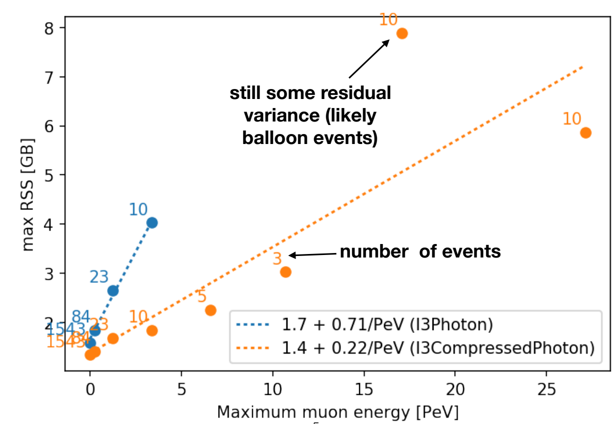
Example: 1e5 events divided into 500 bins (each 1 file)



icecube.weighting.PowerLaw.partition()

4

Example: Gen2 MuonGun simulation

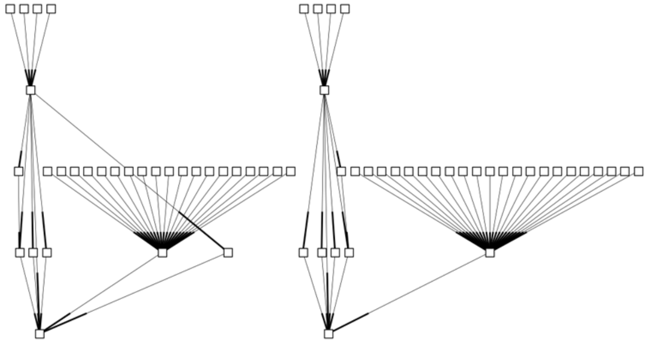


5

*If the failures depend on any feature that propagates to high-level analysis, they bias *any* generation strategy, only in less obvious ways.

A fresh look

- Our operational problem is also an active research topic
- “Knowledge Management in Bioinformatics” group at HU investigates schedulers for e.g. gene sequencing workflows that use black-box resource requirement predictions to reduce makespan
- Use the same ideas to develop an allocation strategy that minimizes resource wastage, and test in on archival IceProd job logs



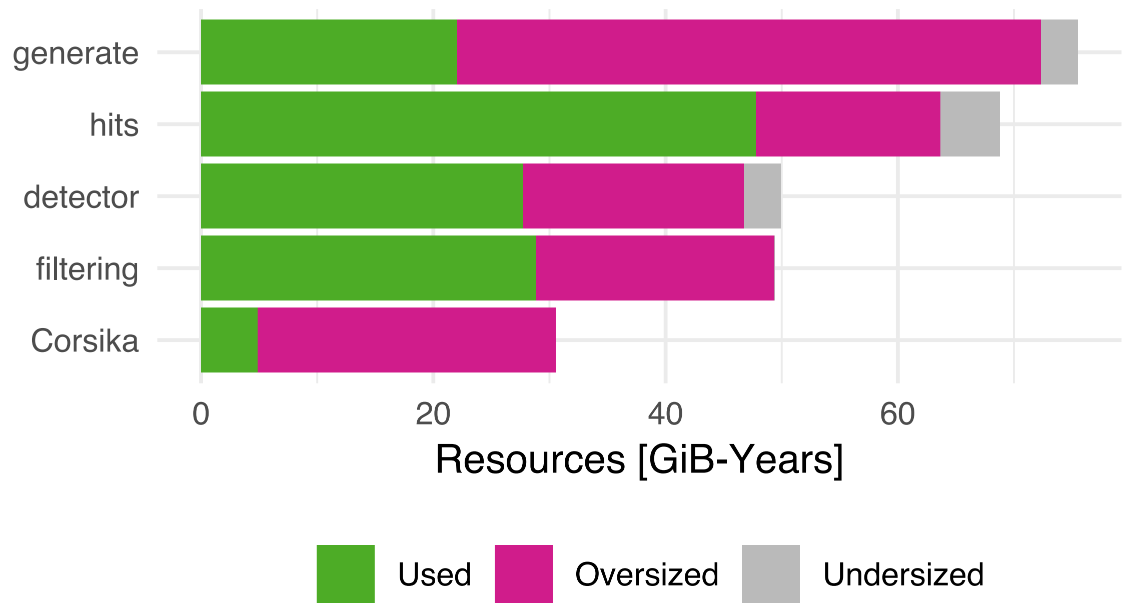
Sipht

(sRNA search workflow)

https://www.informatik.hu-berlin.de/de/forschung/gebiete/wbi

The log data

- 10 months of IceProd job logs (SQL dump provided by D. Schultz)
- 727220 tasks with actual reported memory usage
- 1 PB memory usage
- 76 core-years



Metric: memory allocation quality

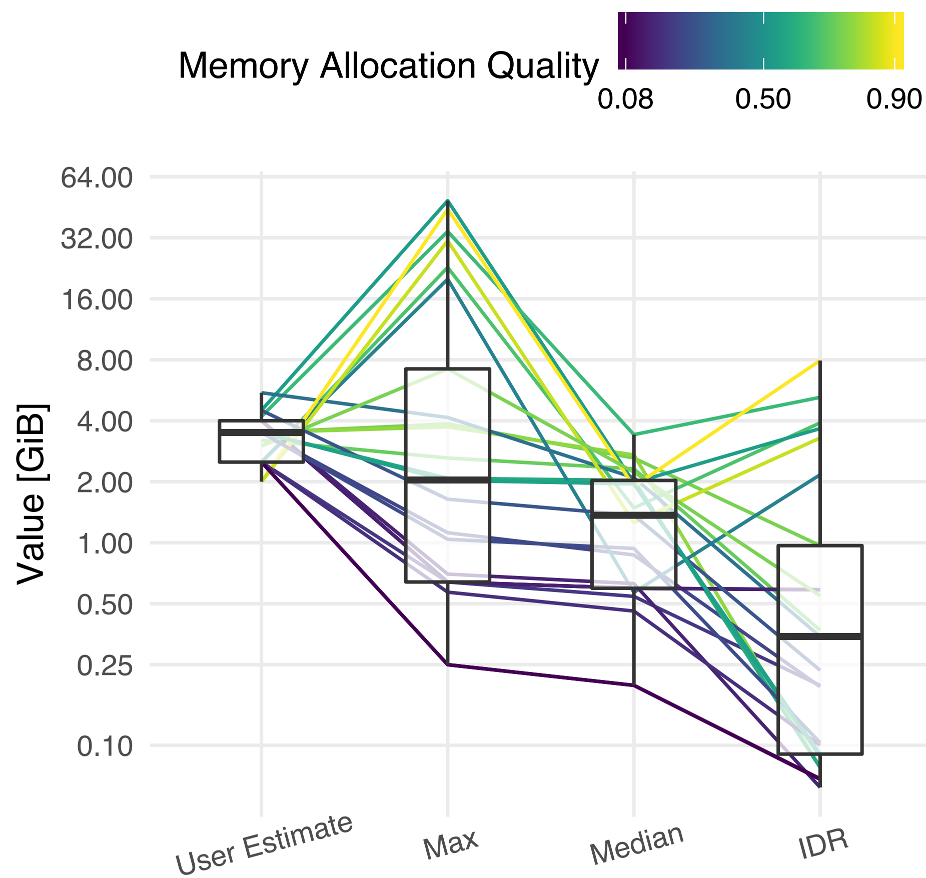
$$\mathrm{MAQ} = \frac{U}{U + W}$$

U: Peak usage*run time

W: wastage*run time (oversizing, failed undersized tasks)

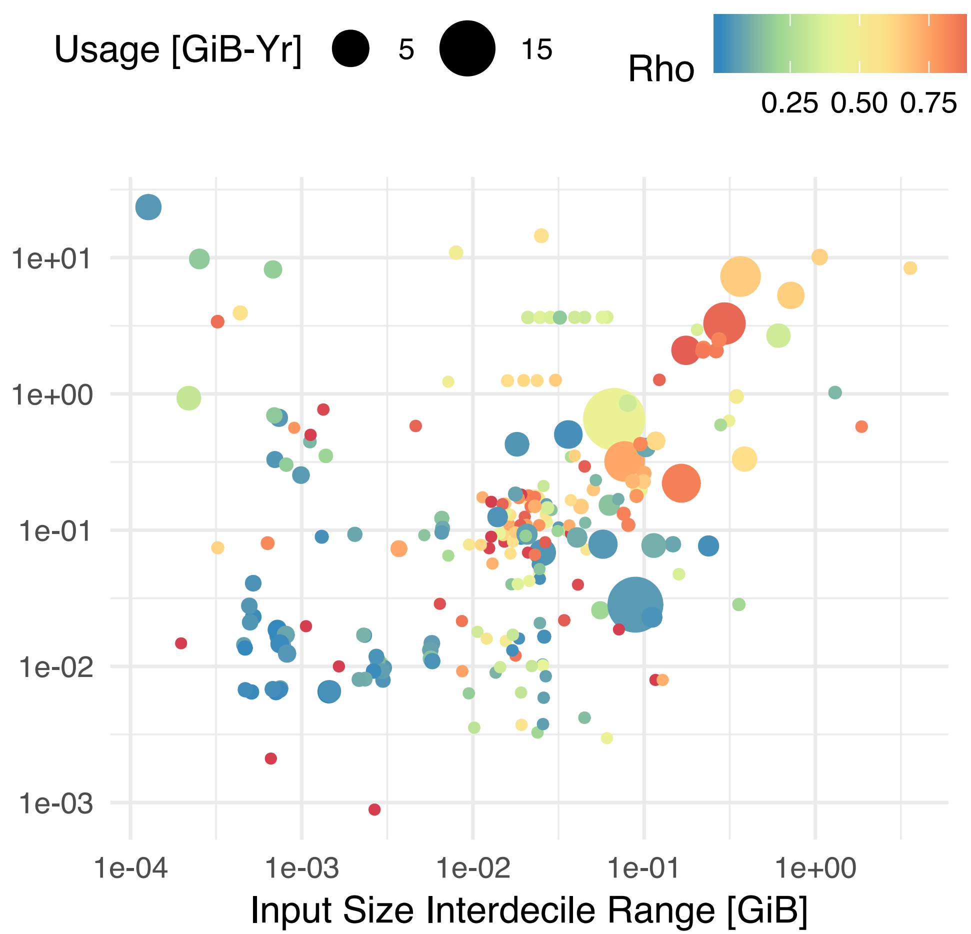
Insights (1)

- Compare initial request, max, median, and inter-decile range for 25 most time-consuming task definitions
- Range of actual max memory usage much larger than initial request
- Worst allocation quality from small, undersized tasks



Insights (2)

- Compare input file size and task memory requirement
- Strong correlation for tasks where both vary strongly
- We can use in the input file size to predict peak memory usage

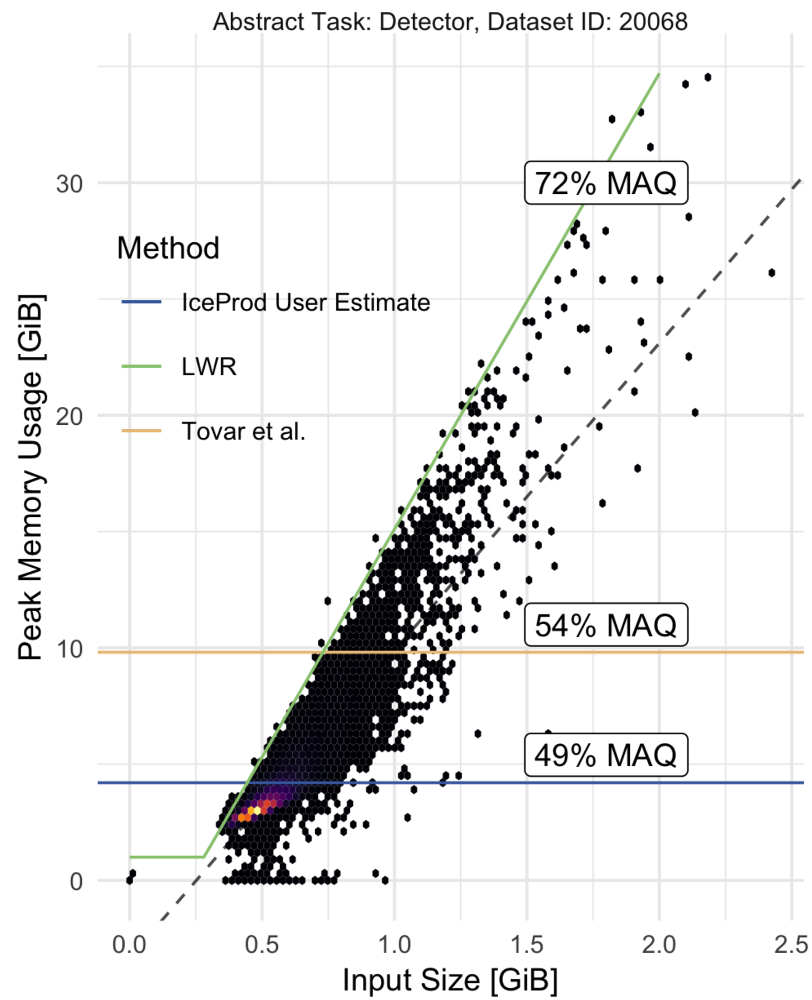


Example

Default IceProd strategy vs. state of the art

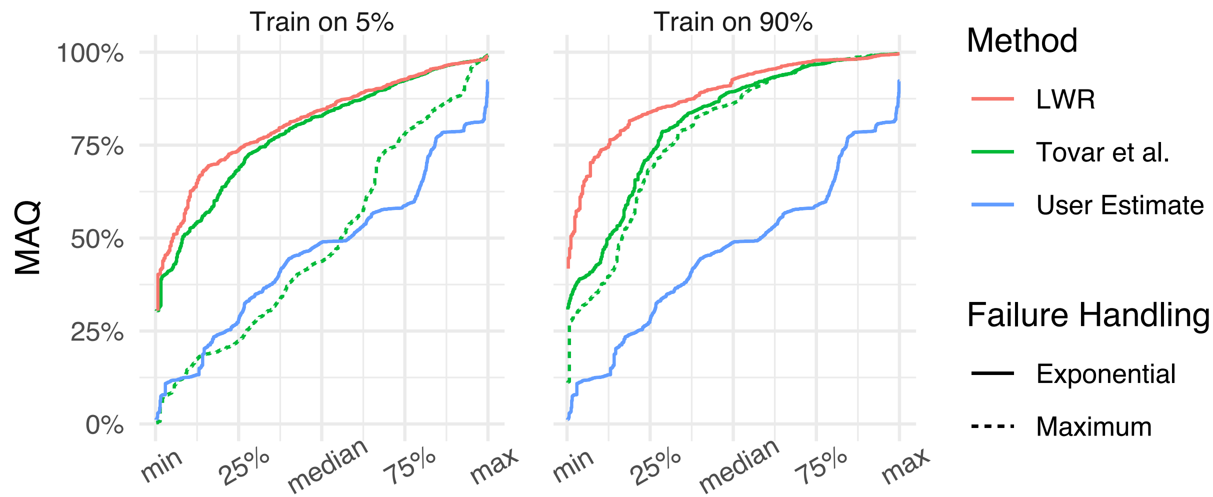
- **User estimate**: start with 4 GB
  - on failure: double request and retry
- **Tovar et al.**: start with 4 GB
  - on first failure: retry with largest memory usage seen so far
  - on second failure: retry with largest possible memory request
- **LWR** (this work): run first 5% of tasks with 4 GB request
  - for subsequent tasks, use linear model on input size; refine model as tasks complete
  - on failure: double request and retry

(NB: not all tasks have this nice of a correlation)

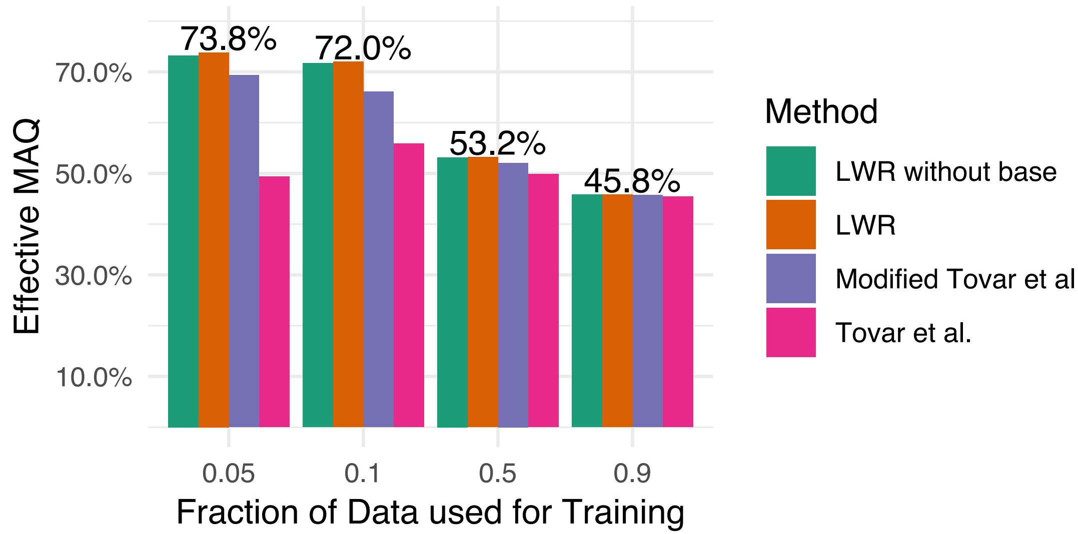


Choosing the size of the training set

Longer training phase improves prediction quality



But: overall MAQ suffers from wastage



DESY.

Summary

- Low-wastage regression can improve memory allocation quality for IceProd jobs by nearly 50%.
- Largest improvement when memory requirement can be predicted from upstream tasks
- **Black-box, online method**: no knowledge of the task content or initial benchmarking needed
- Next steps:
  - Present at HPCS 2019
  - Implement requirement prediction in IceProd2 (who, when?)
  - Gather more log data from newer IceProd2 releases (memory use wasn’t collected for nearly a year)
  - Investigate predictions based on dataset config (i.e. meta project version, generator, number of events, energy range, etc)

Further reading

- B. Tovar, R. Ferreira da Silva, G. Juve, E. Deelman, W. Allcock, D. Thain, and M. Livny, “A Job Sizing Strategy for High-Throughput Scientific Workflows,” TPDS, vol. 29, no. 2, pp. 240–253, 2018.
- Witt, C., Bux, M., Gusew, W. and Leser, U., 2018. “Predictive Performance Modeling for Distributed Batch Processing using Black-Box Monitoring and Machine Learning". /Under review for Information Systems. arXiv:1805.11877/
- Witt, C. van Santen, J., Leser, U., 2019 “Learning Low-Wastage Memory Allocations for Scientific Workflows at IceCube.” HPCS 2019 (contact JvS for a preprint)