Automatic I/O Scheduling Algorithm Selection for Parallel File Systems

Francieli Zanon Boito
UFRGS/UFSC / INRIA Grenoble
Ramon Nou
BSC

This work has received partial funding from the European Union's Horizon 2020 Programme (2014-2020) and from Brazilian Ministry of Science, Technology and Innovation through Rede Nacional de Pesquisa (RNP) under the HPC4E Project (www.hpc4e.eu), grant agreement n° 689772.

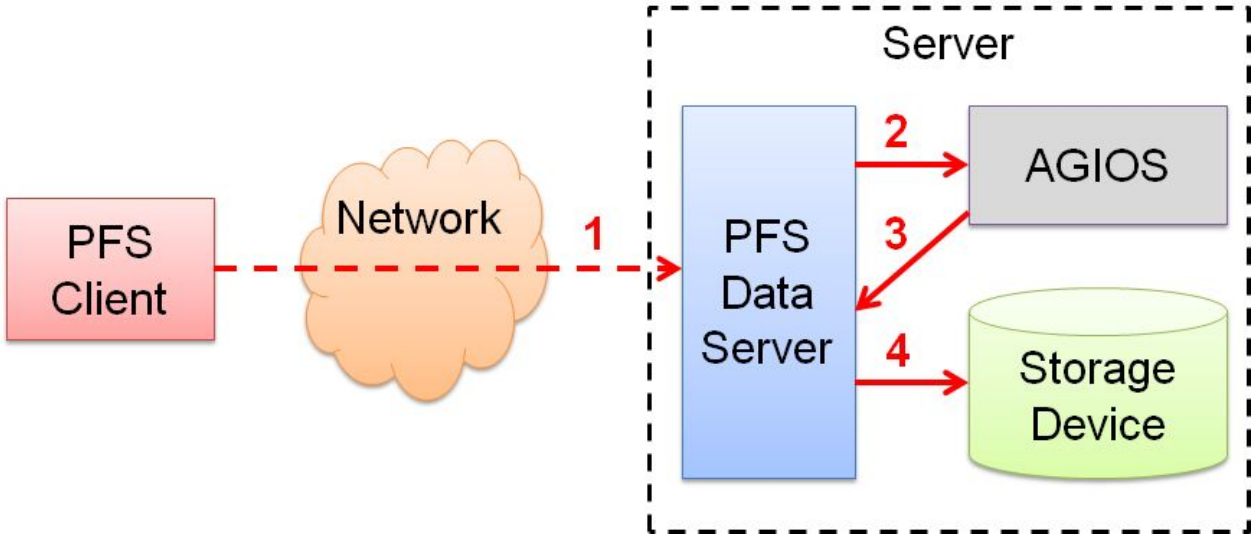
Components of the collaboration

- **AGIOS** : Application-guided I/O scheduler for parallel file systems
  - Selects the best scheduler for an entire workload using decision trees
  - Parallel file system oriented
  - **Requests to files**

- **I/O analyzer** : Automatic I/O scheduler selection through online workload analysis
  - Selects the best scheduler for the next possible part of the workload using markov chains and pattern matching on traces
  - Local devices oriented (kernel level)
  - **Requests to blocks**

AGIOS

- **I/O scheduling library**
- Multiple scheduling algorithms
- Can be used by any I/O service at file level (so far: PFS data servers)
- **Local decisions**

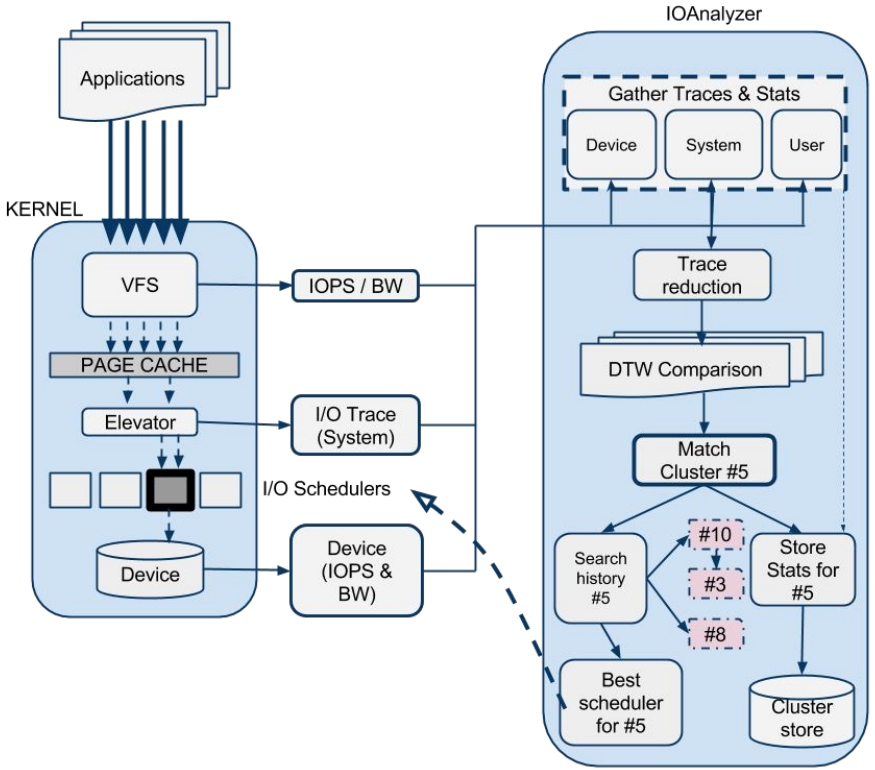


AGIOS

- Selects best scheduling algorithms
- Decision tree
- Performance improvements
  - 75% over aIOLi and 38% over SJF
  - **Increases performance for 64% more situations**
  - **Decreases performance for 89% less situations**
- **Information from traces** (the whole workload)

I/O analyzer

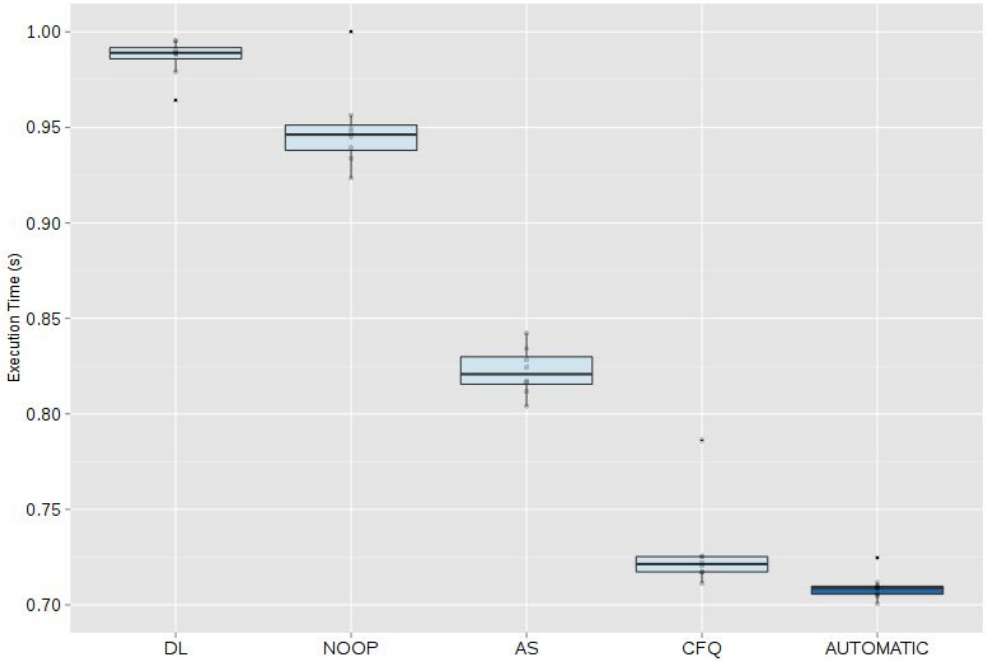
- Captures traces and stats (i.e., each 5 seconds)
- Compares traces with old traces using DTW and heuristics.
- Obtains a match or no, a performance metric and the I/O scheduler used.
  - Updates history
- If there is enough information selects "best" I/O scheduler.



I/O analyzer

- If we do not have a match we use performance to favour a scheduler.

- For example, execution time of 4 parallel processes doing intensive I/O with different I/O schedulers and dynamic.

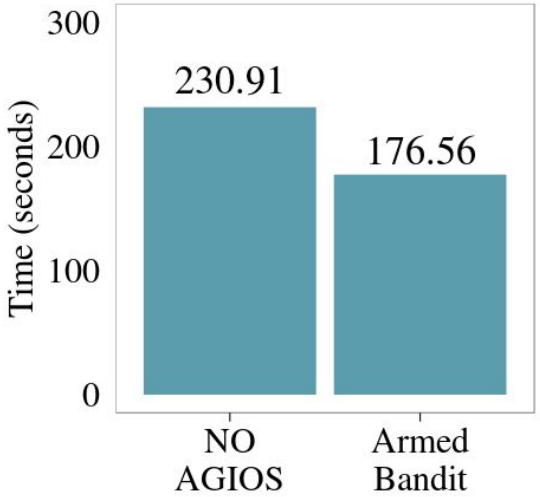


Collaboration

- Study how parallel workloads and AGIOS benefit from intra-workload scheduler changes
  - How long should we wait before issuing a schedule change?
    - **Overhead**
  - Which information can we capture to guide it?
    - **Relative offsets, sizes, timestamps, performance**

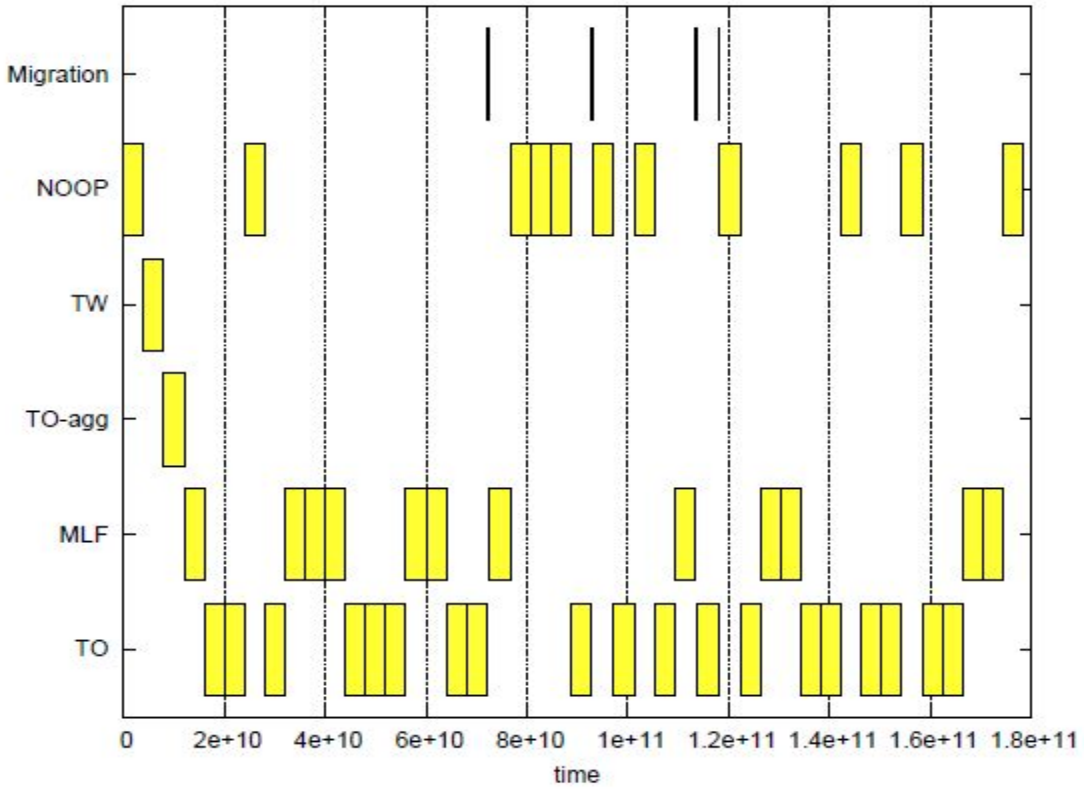
Armed Bandit approach

- Add a naive algorithm for I/O scheduler selection: **Armed Bandit**
  - Each scheduler has selection probability based on the immediate past performance
  - **Performance improvements** to OrangeFS comparable to static algorithms
  - Large tests where one choice is clearly better
  - **Large time windows** (4s)



Armed Bandit approach

- Local decisions



New pattern matching approach

- Adding the **pattern matching and prediction** of I/O Analyzer (work in progress)
- Trace capture and analysis
  - We create a virtual disk where files are sequentially positioned
- Incorporated a markov chain predictor
- **Initial results:** good pattern matching capability
  - 1 pattern generates ~5 patterns
  - next steps: pattern compression

Collaboration

- Slow but continuous progress.
- Scarce resources :)
  - Sometimes collaboration is easier if we have our main project aligned. But it is not the case.
  - Clusters

Questions?

francieli.zanon@inf.ufrgs.br / ramon.nou@bsc.es