Research Group
German Climate Computing Center

Semi-automatic Assessment of I/O Behavior

An Explorative Study on 10^6 Jobs

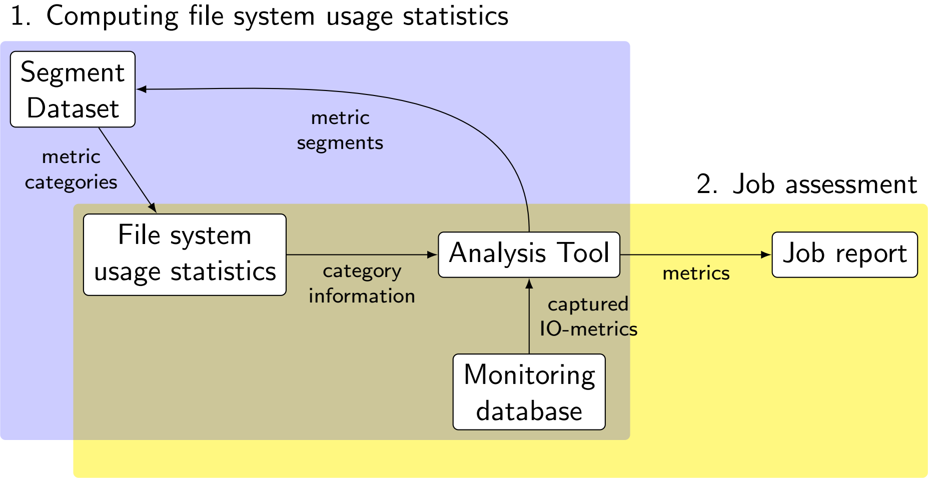
SC19-PDSW
November 18, 2019

Eugen Betke, Julian Kunkel

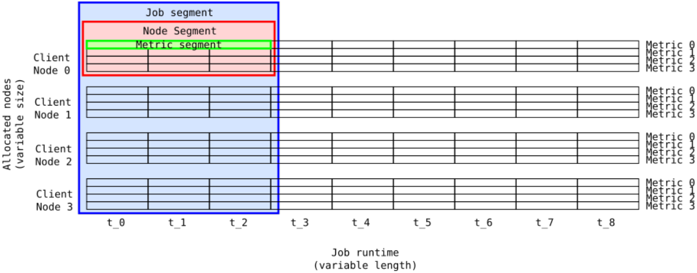
Motivation

- Goals: Finding jobs with
  - high I/O load, but inefficient data access
    - e.g., for application optimization
  - critical I/O load, that affects file system performance
    - e.g., for better job scheduling
- Strategy:
  - Define simple job metrics
  - Use them for ranking and comparison

Analysis Workflow



Segmentation and Scoring of Monitoring Data



Category	Criteria	MScore
LowIO	smaller than q99	0
HighIO	between q99 and q99.9	1
CriticalIO	larger than q99.9	4

Categorization criteria and scores

Score name	Definition
MScore	0,1 or 4
NScore	$\sum$ MScore
JScore	$\sum$ NScore

Segment scores

1. Segmentation
   - Segment size = 3 time points (in this example only)
2. Categorization
   - Quantiles $q99$ and $q99.9$ define thresholds
3. Scoring
   - CriticalIO is at least 4x higher than HighIO

File System Usage Statistics

Metric		Limits		Number of occurences		
Name	Unit	q99	q99.9	LowIO	HighIO	CriticalIO
md_file_create	Op/s	0.17	1.34	65,829K	622K	156K
md_file_delete	Op/s	0.00	0.41	65,824K	545K	172K
md_mod	Op/s	0.00	0.67	65,752K	642K	146K
md_other	Op/s	20.87	79.31	65,559K	763K	212K
md_read	Op/s	371.17	7084.16	65,281K	1,028K	225K
osc_read_bytes	MiB/s	1.98	93.58	17,317K	188K	30K
osc_read_calls	Op/s	5.65	32.23	17,215K	287K	33K
osc_write_bytes	MiB/s	8.17	64.64	16,935K	159K	26K
osc_write_calls	Op/s	2.77	17.37	16,926K	167K	27K
read_bytes	MiB/s	28.69	276.09	66,661K	865K	233K
read_calls	Op/s	348.91	1573.45	67,014K	360K	385K
write_bytes	MiB/s	9.84	80.10	61,938K	619K	155K
write_calls	Op/s	198.56	6149.64	61,860K	662K	174K

Metrics

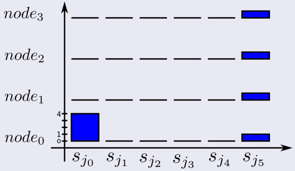
$$\text{Job-IO-Balance (B)} = \text{mean}\left(\left\{\frac{\text{mean\_score (j)}}{\text{max\_score (j)}}\right\}_{j\in\text{IOJS}}\right)$$

$$\text{Job-IO-Utilization (U)} = \sum_{FS}\frac{\sum_{j\in IOJS}\text{max\_score}(j)}{N}$$

$$\text{Job-IO-Problem-Time (PT)} = \frac{\text{count (IOJS)}}{\text{count (JS)}}$$

- FS: Filesystems
- JS: Job segments
- IOJS: IO-intensive job segments

Example



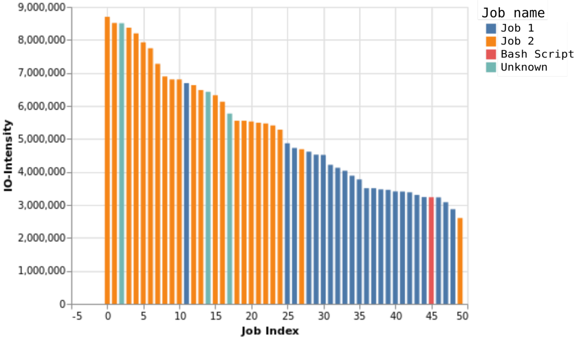
$$\text{Job-IO-Balance} = 0,625$$

$$\text{Job-IO-Utilization} = 2.5$$

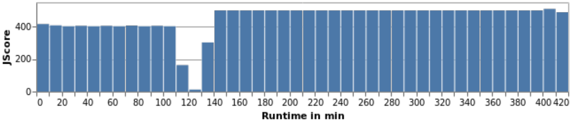
$$\text{IO-Job-Problem-Time} \approx 0.33$$

Jobs with high I/O-Intensity

$$\text{Job-IO-Intensity} = \text{B} \cdot \text{PT} \cdot \text{U} \cdot \text{total\_nodes}$$



30 jobs ordered by IO-Intensity



Nodes: 100; B: 0.88; PT:1.0; U: 4.0

Summary

- Applied methods
  - **Segmentation**: Preserves time line information
  - **Categorization**: Filters not significant I/O and make incompatible metrics compatible
  - **Scoring**: Allows mathematical computation
- Job-IO-Problem-Time, Job-IO-Balance and Job-IO-Utilization
  - Are basic and **simple** metrics
- IO-Intensity and IO-Problem-Score
  - Are a kind of queries, used for **job ranking**