A Workflow for Identifying Jobs with Similar I/O Behavior Utilizing Time Series Analysis

Limitless Storage
Limitless Possibilities

https://hps.vi4io.org

Julian M. Kunkel, Eugen Betke

CHPC Conference

2020-12-02

Copyright University of Reading
Outline

1. Introduction
2. Approach
3. Evaluation
4. Summary
Motivation

- Data center staff are supporting users
  - Optimization of programs
  - Monitoring of (in)efficient usage

- Assume you identified an interesting job
  - Might be particularly inefficient or efficient
  - e.g., via monitoring/tracing or user feedback

- Questions support staff may have
  - Will optimization pay off to other jobs?
  - Is the job a good blueprint for optimization?

- Problem: 100,000 of jobs are executed on a cluster
  - How can we find similar jobs?
DKRZ Monitoring System

**Details**
- **Periodicity:** 10s
- **Record metrics**
  - From /proc
  - 9 aggregated
- **Jobs are linked to the data**

**Mistral Supercomputer**
- **3,340 Nodes**
- **2 Lustre file systems**
- **52 PByte capacity**
- **100+ OSTs per fs**
Monitoring Data of a Job

- Grafana visualization
- Read/write shown
- Metrics supported
  - md_file_create
  - md_file_delete
  - md_read (only)
  - md_mod(ify)
  - md_other
  - read_bytes
  - read_calls
  - write_bytes
  - write_calls
Outline

1. Introduction
2. Approach
3. Evaluation
4. Summary
Identifying Similar Jobs Using Time Series

- Previous work: utilized clustering algorithm(s)
- Today: workflow for supporting the investigation of jobs
- Derived meaningful distance measures
  - Must compare multiple metrics with different units/Range
  - Must handle variable number of nodes and runtime
- Conduct a study on 580,000 jobs (6 months of data from DKRZ)
  - Recorded using DKRZ monitoring system
  - Compute similarity of all jobs to three reference jobs
  - Quantitative analysis: behavioral comparison
  - Qualitative analysis: investigate top 100 similar jobs manually
  - Explored several algorithms
Handle variable number of nodes and runtime

Variable number of nodes
- Calculate statistics across the number of nodes
- Obtain one time line per job metrics

Segmentation across time dimension
- Segment: calculated average across 10 min time interval
- Result is time series of segments
- Earlier attempt: used additional statistics
Compare Multiple Metrics with Different Units/Range

Convert raw data (X Bytes/s or Y opens/s) to categories

- Categorization based on the quantiles of all jobs segments

![Graph showing quantiles of data written](image)

- Categories:
  - 0 = non-IO (< 99% quantile)
  - 1 = HighIO (< 99.9% quantile)
  - 4 = CriticalIO (>)

- Analysis of result shows that the categories are meaningful
  - e.g., 99% quantile is 1 op/s
Algorithms

- Define the pre-processing and distance metrics
- Explored six algorithms that differ in:
  - Aggregation (Each metric independently/together and which dimensions)
  - Coding (quantized/rounded to 17 states, binary)
  - Distance measure (Levenstein, Euclidean)
- Time series based algorithms
  - B-all: binary encode activity (Yes/No) of all metrics into one series
  - B-aggzero: remove subsequent segments of zero activity
  - Q-lev: quantized coding, Levenstein distance
  - Q-native: quantized, Euclidean distance, sliding window
  - Q-phases: extract phase information (metric $\neq 0$ and match)
- Non time series algorithm: Kolmogorov-Smirnov
  - Concatenate individual node data (instead of averaging)
Outline

1. Introduction
2. Approach
3. Evaluation
4. Summary
The Three Reference Jobs (Average across all nodes)

Job-S: Postprocessing

Job-M: 8 hours, 128 Nodes; other metrics == 0

Job-L: 66 hours, 20 Nodes; other metrics == 0
Introduction

Approach

Evaluation

Summary

Histogram of Jobs in Quantized coding

Job-S Histogram

Job-M Histogram
Similarity to Job-S

- 100% sim is the reference job
  - Job-S is CMORization control
- Names in the job pool
  - 22,580 have somewhat "cmor"
  - 367 have somewhat "control"
- Observations
  - Most jobs have a low similarity
  - Clusters are visible
  - 30-40 of most similar jobs have control in the name
  - All algorithms work somewhat
- User may explore from most similar job to least
  - A cluster is reasonable cut-off
Inspecting Selected Jobs: B_aggzeros

Non-cmor job: Rank 76, SIM=69%

Non-control job: Rank 4, SIM=81%
Inspecting Selected Jobs: Q_Lev

Rank 2, SIM=96%
That looks rather similar, even better than B_aggzeros

Rank 15, SIM=90%

Rank 100, SIM=79%
Similarity to Job-M

- Different behavior than for Job-S
- Much smaller similarity
- Q-native has highest similarity
Inclusivity and Specificity for Job-M 100 Similar Jobs

- The algorithms identify a wide range of job runtime, node counts and different users.

Distribution of node counts (job=128)

Distribution of Runtime (job = 28,828s)
Inspecting Selected Jobs: Q-native

Rank 2, SIM=99%

Rank 3, SIM=97%

Rank 15, SIM=91%

Rank 100, SIM=88%
There are not many such long jobs

The Q-phases and KS algorithms finds many jobs
  ▶ There is only one IO phase
  ▶ Many jobs have indeed similar profiles

What to include?
  ▶ It depends on the definition of similarity
Outline

1 Introduction
2 Approach
3 Evaluation
4 Summary
Conclusions

- Distance measures allow to find similar jobs
- Utilizing time series of system statistics seems suitable
  - All algorithms worked rather well on Job-S
  - For Job-M and Job-S, we prefer Q-native and Q-lev
  - Runtime (not shown here) is also feasible (near realtime)
- It all depends on our expectation of "similarity"
  - What does a user need? Find similar phases in jobs? Do we require the same job length?
  - The community should define "similarity"
- Check the previous paper for this work:
  - Eugen Betke and Julian Kunkel
  - Classifying Temporal Characteristics of Job I/O Using Machine Learning Techniques