Improving System Performance and Tradeoffs via Design and Configuration Space Exploration

Sponsor: DASD(SE)

By
Chong Tang
5th Annual SERC Doctoral Students Forum
November 7, 2017
FHI 360 CONFERENCE CENTER
1825 Connecticut Avenue NW
8th Floor
Washington, DC 20009

www.sercuarc.org

Overview

- Problem
- Goal
- State of Art and Gaps
- Approaches
  - —Formal design space synthesis and exhaustive dynamic tradeoff analysis
  - —Efficient meta-heuristic search for high-performing software configurations
- Evaluation
  - —Experimental case study: Object-Relation Mapping Systems
  - —Experimental case study: Hadoop Big Data Stack Configuration
- Future Work

Problem

Problem

We lack concept, methods, and tools for searching large spaces of complex system designs and configurations to improve system performance and tradeoffs in multiple dimensions absent simple objective functions or knowledge of their mathematical properties and where sampling objective experimentally is costly

Examples

- Find configurations for a nuclear fusion plasma containment vessel with thousands of configuration parameters to increase temperature and extend confinement time
- Find configurations of hardware caches that configure themselves to perform well on diverse application loads
- Synthesize code snippets to pass all tests in given test suites as a mechanisms for enabling software to repair itself
- Find configurations for complex software and cyber-physical systems to improve performance in relevant dimensions

Goal

Research Goal

- Develop and evaluate concepts, methods, and tools ...
- for searching large spaces of complex structures ...
  - Hundreds or thousands of parameters
  - Often not numerical, but values of complex types
  - Parameterized and with dependent structures
- to improve system performance and tradeoffs ...
  - In many dimensions of performance (c.f., Boehm/Sullivan)
- absent mathematically straightforward objective functions ...
- absent knowledge of their mathematical properties
  - E.g., convexity, linearity, etc
- where sampling the objective functions is expensive
  - Where evaluation requires measurement of running system
  - E.g., takes minutes to weeks to evaluate performance for sample

State of the Art and Gaps

Brief Partial Overview

- Self-Tuning Caches
- Program Repair
- Plasma Fusion Reactor Configuration
- Embedded System Design
- Software System Configuration

Self-Tuning Caches [Zhang et al. 04]

- **Problem:** Unnecessary flushing and energy waste
- **Solution:** On-chip design space exploration to tune to application
- **Strength:** searches 4-parameter space in hardware vs. just one
- **Relationship:** dozens or hundreds of more complex parameters

Program Self-Repair [Weimer et al. 09]

- **Problem:** Debugging software remains costly manual process
- **Solution:** Genetic programming to find test-passing code snippets
- **Strengths:** Compelling proof of concept that broke new ground
- **Relationship :** Unsound, effectiveness depends on test suite

Plasma Containment [Baltz et al. 17]

- **Problem:** 1000+ parameters, nonlinearity, unknown objective fn
- **Solution:** *Human-guided* meta-heuristic search over *meta-params*
- **Strength:** Achieved *doubling* of performance in temp and time
- **Relationship:** Similar dimensionality, time-scale, meta-heuristics; different application domain; not exploring human-in-loop; PhD student colleague applying approach to human-robot systems

Embedded Systems [Mohanty et al 02]

- **Problem:** Energy in heterogeneous embedded system design
- **Solution:** Constraint-driven, multi-resolution, simulation-based DSE
- **Strengths:** Low-resolution simulation evaluates designs quickly, while high-resolution evaluators help select designs for production
- **Relationship:** Simulation vs. dynamic profiling; we also address cost of search when high-accuracy sampling of objective fn. is very costly

Software Configuration [Nair et al 17]

- **Problem:** Models to predict performance of configurations costly
- **Solution:** Train models that only predict performance *ordering*
- **Strengths:** 10X reductions in # of samples of objective function
- **Relationship:** We address much larger configuration spaces, 100X reduction in sampling cost yet often finding good configurations

Two Approaches

Approach #1: Formal Synthesis

- Design and embed system modeling language within tractable formal logic with associated general analyzers/synthesizers
  - —E.g., AlloyOM: A language for object-oriented data models, in Alloy [cite]
- Formalize and verify design/configuration spaces as semantics of modeling languages, then use analyzers to synthesize designs
  - —E.g., "mapping rules" from AlloyOM models to corresponding SQL schemas
- Separate, scalable, dynamic analysis of multiple system qualities
- A formal, generalized tool architecture and implementation for
  - —Coq specification and Scala implementation of parameterized framework
  - —Spark-based back-end for arbitrary parallelization of dynamic evaluation

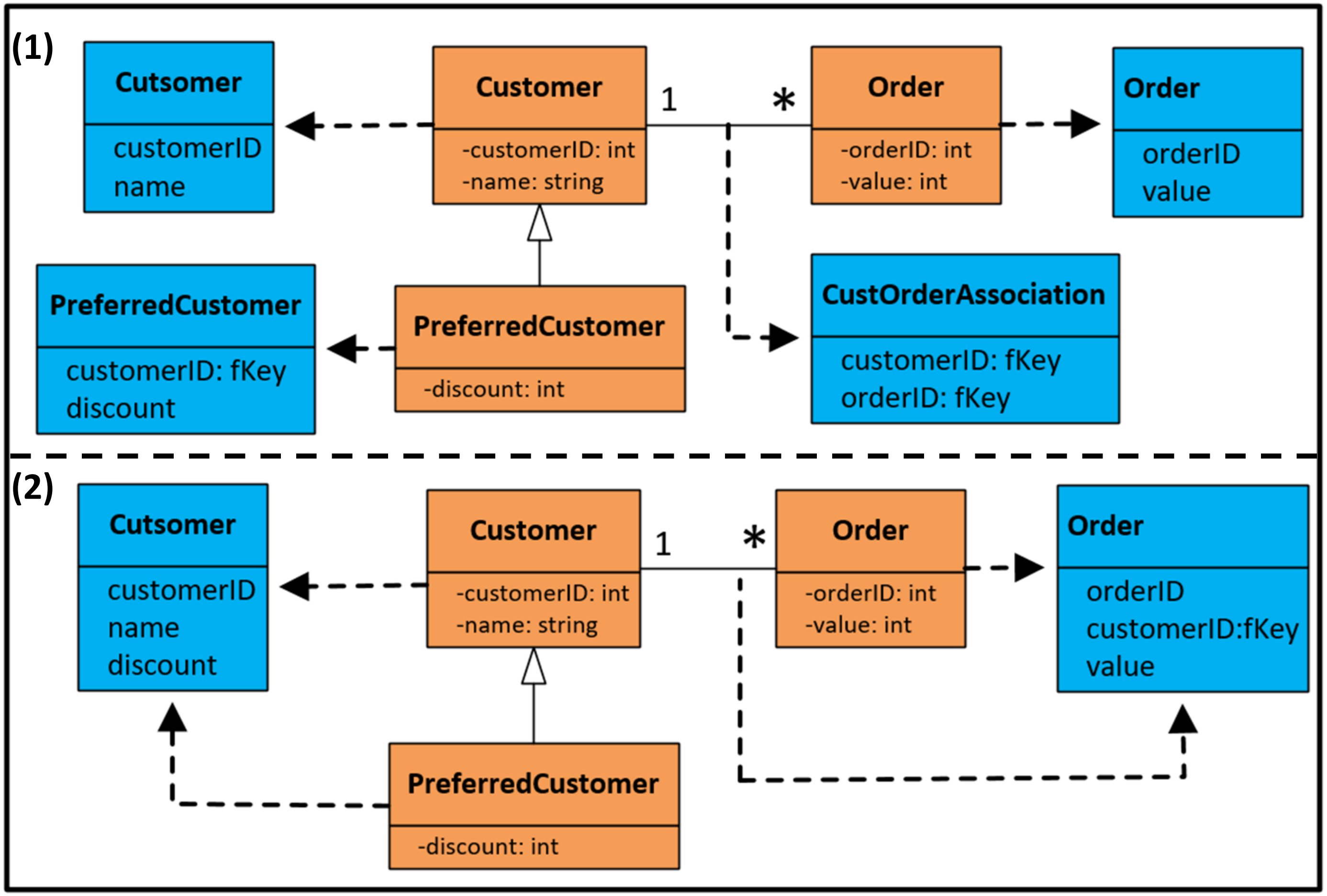
Approach #2: Meta-Heuristic Search

- New focus on large, multi-component, software-intensive systems
  - —E.g., Hadoop stack: HDFS, map-reduce, Yarn, Spark, Hive, etc.
  - —Thousands of parameters of diverse types across many modules
  - —Sampling objective function on production loads is very expensive
- Hypothesis 1: meta-heuristic search in lieu of learning generalized prediction models (e.g., Menzies 17) can be effective
  - —General-purpose tool framework for problems of this kind
    - o Spreadsheet representations of configuration spaces and sampling strategies
    - o Pluggable search algorithms, system set-up, performance measurement, etc.
    - o Facilities for probing local properties of design spaces (e.g., sensitivity analysis)
- Scale-up hypothesis: Evaluating objective function on small instances will (sometimes!) find configurations that achieve significant performance improvements for much larger jobs

Evaluation

#1: Formal Synthesis for ORM

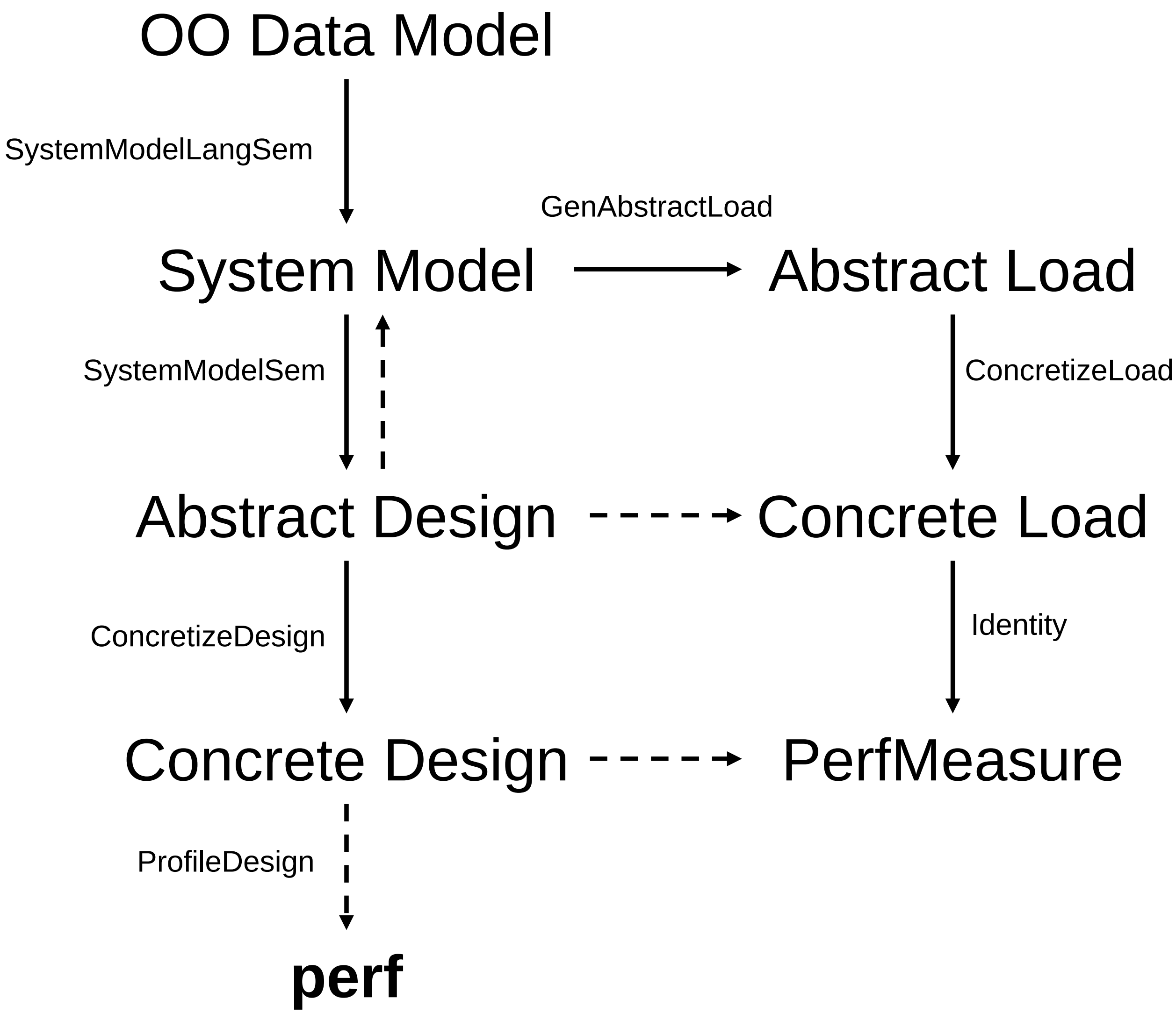
Example: SQL Schema Design



One model

Multiple Designs

Design Space Profile Framework



OO Data Model

SystemModelLangSem

GenAbstractLoad

System Model

Abstract Load

SystemModelSem

ConcretizeLoad

Abstract Design

Concrete Load

ConcretizeDesign

Identity

Concrete Design

PerfMeasure

ProfileDesign

perf

Framework Instantiation

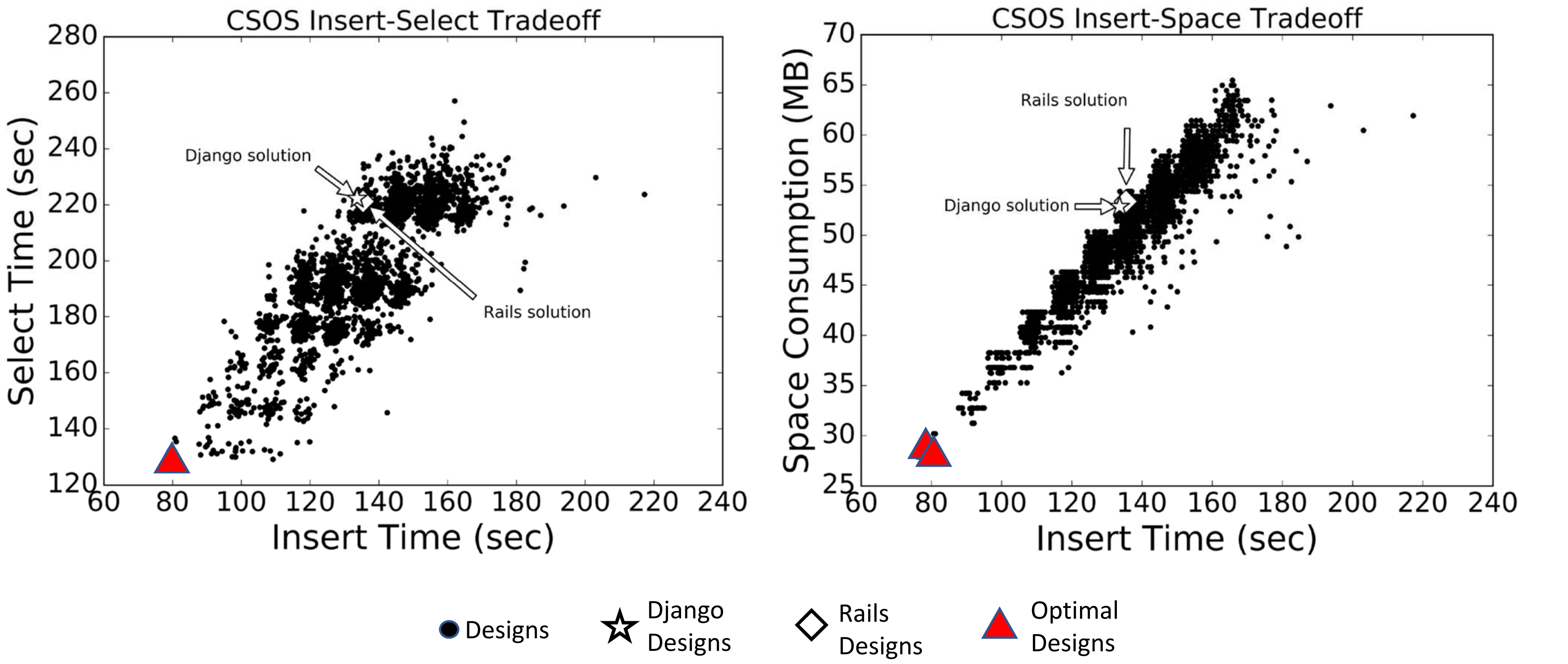
- Domain Specific System Modeling Language embedded in Alloy
- Semantics of system models formalized in Alloy
- Synthesizer based on Alloy Analyzer (relational logic model finder)
- Full design space and test suite synthesis
- Scalable dynamic evaluation framework based on Spark
- Performance vector: <Insertion time, Selection time, Disk space>
- Pareto-optimal front selector and visualization

Experiments Setup

- Compare with Rails and Django
- Study objects: 7 ORM models
- Experiment platform: A Spark cluster
- MySQL and PostgreSQL
- Run each test case 3 times to get the average performance

Hypothesis: Our approach can find SQL schemas with significant better ***time*** and ***space*** performance than those created by *Rails and Django*

Preliminary Results on MySQL



Interpretation

- System formalization and synthesis has the potential to produce much better designs in the domain we have studied
  - — More than 40% improvement in all three performance dimensions
- Threats to validity: generalizability to others domains is untested
- Limitations: doesn’t represent system behavior explicitly
  - —Garlan’s work *[Camera et al. 17’]* includes behavior specification/synthesis
  - —so far been used only to synthesize very small numbers of designs (< 10)
- Future work:
  - —Evaluate our approach on other DB platforms like PostgreSQL
  - —Generalize our approach and framework in other domains

#2: Meta-Heuristic Search for Hadoop Configurations

Motivation

- Hadoop widely used for very costly industrial computations
- Hadoop stack has nearly 1000 diverse configuration parameters
- Research suggests most engineers just use default configurations
- Hypothesis: automated search can significantly reduce costs

Research Questions

- RQ #1: Can meta-heuristic search find better configurations?
- RQ #2: Can we make search affordable by sampling objective function using small jobs as proxies for production-scale jobs?
- RQ #3: What are the computational and monetary costs of this approach and how do they relate to any benefits obtained?

Tiny Example of Hadoop Configuration

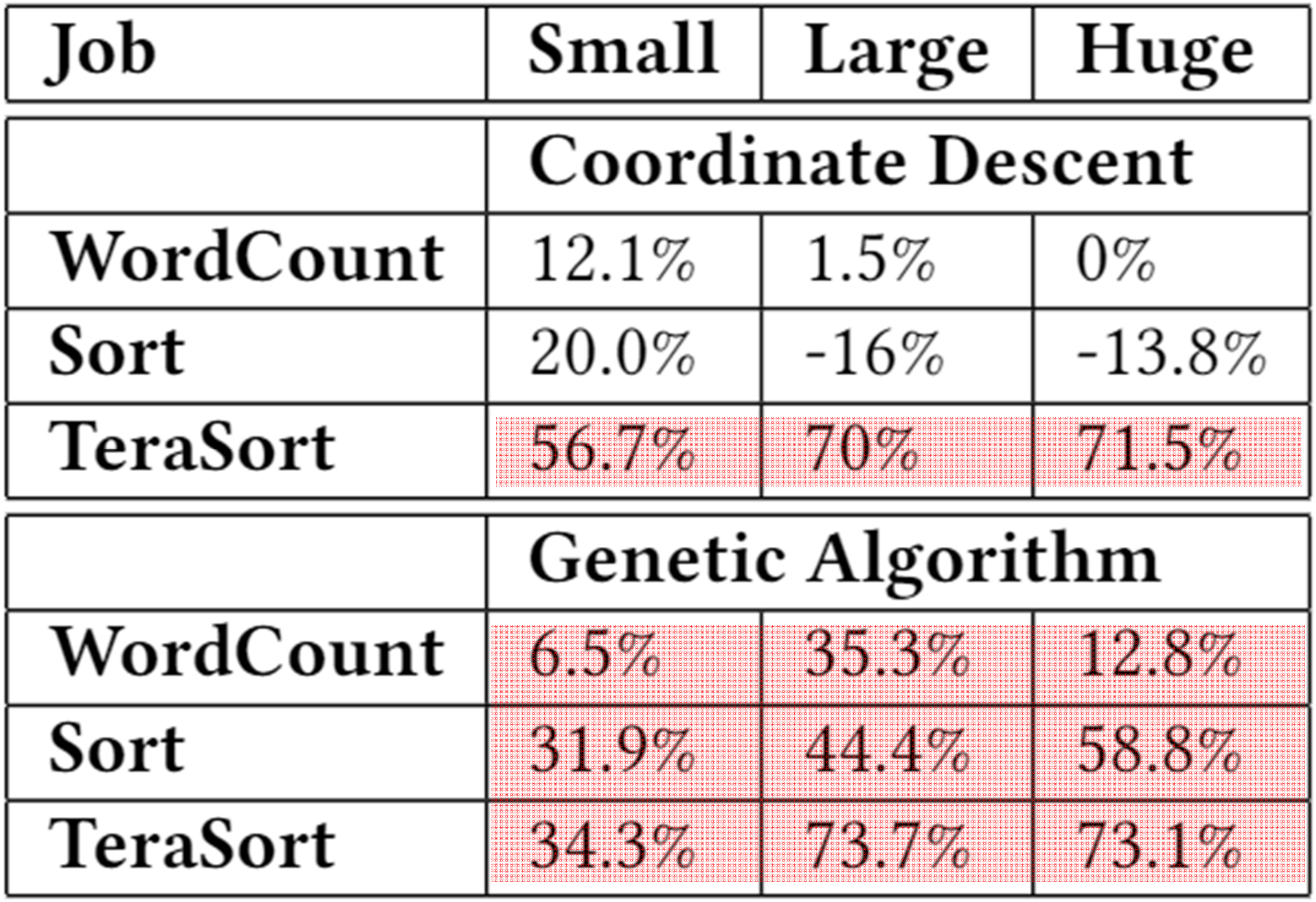
Parameters	AWS Default	GA Best Value
mapreduce.reduce.input.buffer.percent	0	0.4
mapreduce.map.output.compress	True	False
yarn.resourcemanager.scheduler.class	CapacityScheduler	FairScheduler
mapreduce.map.cpu.vcores	1	5
mapreduce.reduce.cpu.vcores	1	4

Framework

- Configuration space, sampling space encoded in Excel
- Coordinate descent and genetic algorithms evaluated
- Automated generation, set-up, and profiling of Hadoop stacks
- Profile instances using HiBench loads on AWS cluster
  - —5-node AWS EC2 m4.xlarge instances
  - —HiBench “small” (320MB) datasets for several job type
- Measure CPU time (and map to $ costs)
- Validate resulting configurations on large (32GB) loads
- Sensitivity analysis on changes from default configuration

Preliminary Results

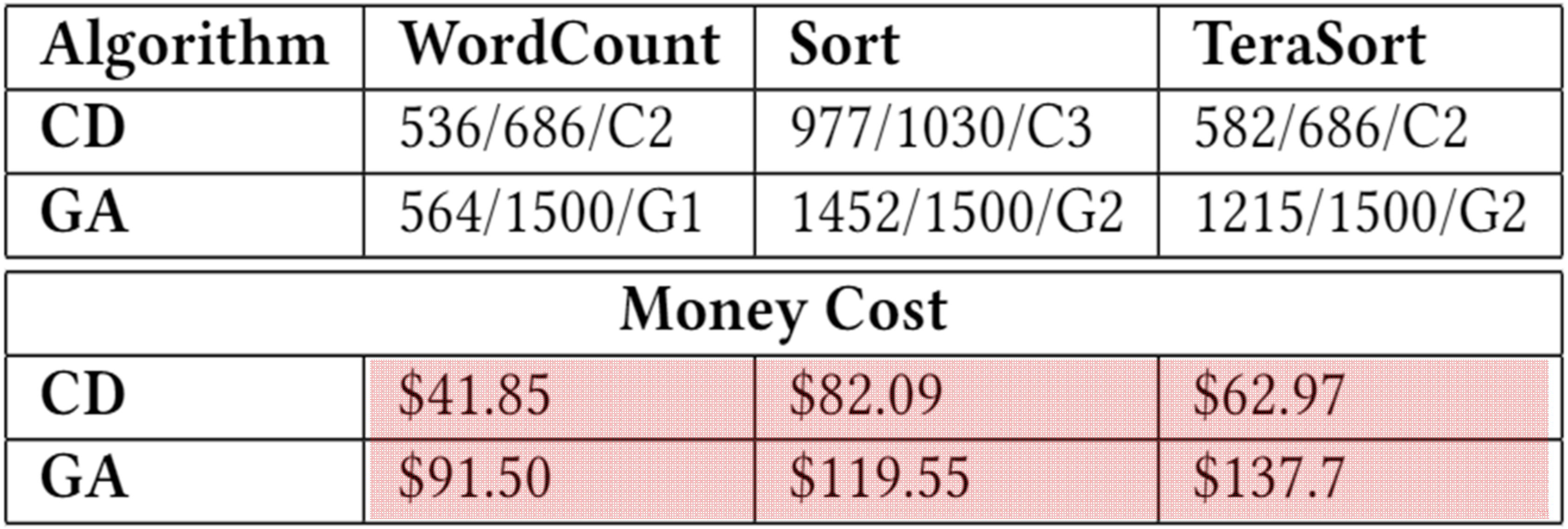
--- CPU time improvement



Job	Small	Large	Huge
	Coordinate Descent		
WordCount	12.1%	1.5%	0%
Sort	20.0%	-16%	-13.8%
TeraSort	56.7%	70%	71.5%
	Genetic Algorithm		
WordCount	6.5%	35.3%	12.8%
Sort	31.9%	44.4%	58.8%
TeraSort	34.3%	73.7%	73.1%

Preliminary Results

--- Algorithm performance & Cost



Algorithm	WordCount	Sort	TeraSort
CD	536/686/C2	977/1030/C3	582/686/C2
GA	564/1500/G1	1452/1500/G2	1215/1500/G2

Money Cost			
CD	$41.85	$82.09	$62.97
GA	$91.50	$119.55	$137.7

Interpretation

- Meta-heuristic search shows some promise for hiding high-performing configurations in important, high-dimensional spaces
  - 30% - 70% CPU time improvement for “large” and “huge” loads
  - Cost-effective: $100 or less on AWS to find resulting configurations
  - Small jobs as “low-resolution” proxy for objective function worked “pretty well”

- Limitations and future work:
  - —Generalizability of results beyond certain classes of Hadoop jobs unclear
  - —Improve performance of other systems like Spark
  - —Represent configuration spaces with formal language-based methods
  - —Evaluate performance in multiple dimensions
  - —Can sensitivity analysis reveal modular structure in parameter space?

Project Plan

Future Work Plans

- **#1 Formal Synthesis:**
  - —*Extend evaluation for ORM using PostgreSQL DBMS*
  - —Develop and evaluate applications beyond ORM domain

- **#2 Meta-Heuristic Search**
  - —*Leverage constraints, hierarchy, dependent typing to reduce search space*
  - —*Map modularity in parameter space, if any, to simplify search problem*
  - —Incorporate generalized taxonomy of system qualities into framework
  - —Human-in-loop configuration for IoT, CPS, and AI-driven “Centaur” systems

Conclusion

- "Software is eating the world" –M. Andreessen
- Configuring software-intensive systems increasingly difficult
- Only going to get worse in a world of IoT, autonomy, CPS, AI, etc.
- Two approaches to "optimizing" software configurations
  - Formal languages and synthesis + scalable performance and tradeoff analysis
  - Meta-heuristic search with small-job proxies for full-blow objective functions
- Early results suggest that such approaches are worth pursuing
  - 40% time and space improvement for ORM vs widely-used tools
  - 30% - 70% reduction in CPU time for important classes of Hadoop jobs

Bibliography

- [Zhang et al. 04] Zhang et al, C., Vahid, F., & Lysecky, R. ***A self-tuning cache architecture for embedded systems***. [*DATE 04'*]

- [Weimer et al. 09] Weimer, W., Nguyen, T., Le Goues, C., & Forrest, S. ***Automatically finding patches using genetic programming***. [*ICSE 09'*]

- [Baltz et al. 17] Baltz, E. A., Trask, E., Binderbauer, M., Dikovsky, M., Gota, H., Mendoza, R., … Riley, P. F. ***Achievement of Sustained Net Plasma Heating in a Fusion Experiment with the Optometrist Algorithm***. [*Nature 17'*]

- [Mohanty et al. 02] Mohanty, S., Prasanna, V. K., Neema, S., & Davis, J. ***Rapid design space exploration of heterogeneous embedded systems using symbolic search and multi-granular simulation***. [*LCTES 02'*]

- [Nair et al. 17] Vivek Nair, Tim Menzies, Norbert Siegmund, and Sven Apel. ***Using bad learners to find good configurations***. [*FSE 17'*]

- [Camera et al. 17] Cámara J, Garlan D, Schmerl B. ***Synthesis and Quantitative Verification of Tradeoff Spaces for Families of Software Systems***. [*ECSA 17'*]

Thank you!

Questions?