Enhancing the Configuration Tuning Pipeline of Large-Scale Distributed Applications Using Large Language Models (Idea Paper)

Gagan Somashekar*, Rajat Kumar*

*equal contribution
Introduction

• Microservices architecture is replacing monolithic or multi-tier architecture

• Performance is crucial as these are usually customer-facing applications

Image source: Gan et al., ASPLOS’19
Motivation

• Performance depends on the configuration of the application (here, social networking application)

![Graph showing performance differences between configurations C1, C2, and C3.]

- C1 – only worker_process tuned
- C2 – all parameters jointly tuned
- C3 – all parameters tuned but worker_process assigned a bad value
Motivation

• The first-step of configuration tuning – parameters’ meta-data extraction
  • name, range, default, dependencies, etc.

• A stage that should be revisited
  • Application architecture
  • Software updates – parameter addition, deprecation
  • Deployment and hardware changes
Motivation

• **Very large** configuration space.
  • $n$ – number of microservices
  • $p$ – parameters per microservice
  • $c$ – number of configurations per parameter
  • **Total possible configurations** $\approx c^{n*p}$

• Parameter dependencies are crucial for **reducing** configuration search space
  • Absolute – Redis’ “maxmemory” and “maxmemory-policy”
  • Partial – Redis’ “maxmemory” and container’s “mem-limit”
  • Performance – MongoDB’s “concurrent_reads” and “cache_size”
Motivation

• The meta-data of the parameters is found in
  • Product manuals, blogs, etc.
  • Source code and documentation

• Experimental feedback necessary to ascertain certain meta-data
  • Nginx “threads” and “max_queue” parameters

• A practitioner “understands” crucial information in the product manuals and, guided by empirical observations and telemetry, tunes the application to obtain optimal results

Can automatically extracted meta-data be coupled with experimental feedback to enhance configuration tuning pipeline in large-scale distributed applications?
Related Work

• DB-BERT
• SafeTune
• SPEX
• Prior works don’t utilize the full potential of NLP as they don’t:
  • Perform fully automated and exhaustive mining of text
  • Utilize language models for learning new associations and dependencies based on experimental feedback.
Proposal

Large Language Models (LLMs) for enhancing the configuration tuning pipeline!

- Meta-data extraction using a targeted language model
- Enhance the configuration tuning pipeline using the LLM
- In-house knowledge system
Envisioned Pipeline

1. Parameter Meta-data
2. Tuning Algorithm
3. Optional Training
4. Application 1
5. Workload
6. Experimental Data and workload characteristics
7. Prompt Generator

Product Manuals, etc.
LLMs in Configuration Tuning

Layer Norm

Feed Forward

Layer Norm

Masked Multi Self Attention

Text & Position Embed

Pretrained Language Model

Domain Adaptation

Gradual Task Adaptation

Domain Adapted Language Model

Product Manuals, Developer articles, etc.

Experimental Data
LLMs in Configuration Tuning

• Domain adaptation
  • Mitigate against domain shift

• Prompt engineering
  • “The default value of ldapUserCacheStalenessInterval is”

• Building an in-house knowledge system
  • {
    "prompt": "<verbal description of the workload and the architecture of the application >",
    "completion": "<optimal subset of parameters>"
  }
Planned Evaluation and Conclusion

• Quality of meta-data generated
• Developer hours saved
• Quality of impactful parameters
• Generalization