Rhythm: Component-distinguishable Workload Deployment in Datacenters

Laiping Zhao\textsuperscript{1}, Yanan Yang\textsuperscript{1}, Kaixuan Zhang\textsuperscript{1}, Xiaobo Zhou\textsuperscript{1}, Tie Qiu\textsuperscript{1}, Keqiu Li\textsuperscript{1}, Yungang Bao\textsuperscript{2}

\textsuperscript{1}Tianjin University, \textsuperscript{2}Inst. Of Computing Technology, CAS
Outline

- Background
- Interference on LC components
- Rhythm Controller
- Experimental Evaluation
- Conclusion
Background

- Low Resource Utilization of Datacenter

![Graph showing CPU utilization over time]

- Aliyun: The average CPU utilization of co-located cluster approaches to **40%** [Guo, 2019].

- Improved, but still low utilization.
Background

- **Co-location**: Improving the resource utilization
- **Interference** causes **unpredictable latency**.

- Profiling of the workload.
- Schedule in a cross-complementing way.
- **Real-time monitoring**;
- **Passive adjustment on resource allocation**.
Background

- Many-component Services:
  - SocialNetwork service.
  - 31 microservices.
  - A Single Transaction Across ~40 Racks of ~60 Servers Each.
  - Arc: client-server RPC.

Source: [Deathstarbench, ASPLOS’19]

Source: [Google, Datacenter Computers modern challenges in CPU design, 2015]
Problem

How can we feedback-control when a request is served by multiple components collaboratively?

- Latency: \( L_{overall} = L_{cpn1} + L_{cpn2} + \cdots \)
- Tail Latency: \( TL_{overall} = f(TL_{cpn1} + TL_{cpn2} + \cdots) \)

Given an overall TL, how to derive a sub-TL for each component?

OR: How the component-control affect the overall-TL?
Inconsistent Interference Tolerance

Components perform significant difference (~435%) under the same source of interference.
Rhythm Design

Rhythm Insight:

- **Components with smaller contributions to the tail latency can be co-located with BE jobs aggressively.**

Challenges:

- **How to quantify the contributions of a component?**
- **How to control the BE deployment aggressively?**
  - *When to colocate?*
  - *How many BEs can we co-locate with the LC?*
Inconsistent interference tolerance ability;

Tracking user request:
Request tracer

- Causal path graph
- Send/Receive events: ACCEPT, RECV, SEND, CLOSE
- Event: `<type, timestamp, context identifier, message identifier>`
- Context: `<hostIP, programName, processID, threadID>`
- Message: `<senderIP, senderPort, receiverIP, receivePort, messageSize>`
Inconsistent interference tolerance ability;

Tracking user request;

Servpod abstraction:

- A collection of service components from one LC service that are deployed together on the same physical machine.

- For deriving the sojourn time of each request in each server.
- Inconsistent interference tolerance ability;

- Tracking user request;

- Servpod abstraction;

Contribution analyzing:

- Contribution of servpod1
- Contribution of servpod2
- Contribution of servpod3
Servpods with higher average sojourn time contribute more to TL.

Servpods with higher sojourn time variance contribute more to TL.

Servpods that highly correlated with the tail latency contribute more to tail latency.

\[ C_i = f(\rho_{T_i, T_{99th}}, P_i, V_i) = \rho_{T_i, T_{99th}} P_i V_i \]
Is this definition effective?

Sensitivity vs contributions

The increase in the 99th-tile latency when a single Servpod is interfered by different BEs:

- Mixed BEs of wordcount, imageClassify, lstm, CPU-stress, stream-dram and stream-llc.
- DRAM intensive: Stream-dram
- CPU intensive: CPU-stress
- LLC intensive: Stream-llc.
- Inconsistent interference tolerance ability;
- Tracking user request;
- Servpod abstraction;
- Contribution analyzing;

Controller:
- **Loadlimit**: allowing colocation when $\text{load} < \text{loadlimit}$;
  - The “Knee point” of performance-load curve.

- **Slacklimit**: the lower bound of slack for allowing the growth of BEs.
  - Slack = $SLA - currentTL$;
  - Small contribution $\rightarrow$ larger slacklimit;
Controller

- **When can we co-locate workloads?**

  - Loadlimit.

- **Loadlimit** per servpod:

  - The upper bound of the request load for allowing the colocation with BE jobs;

  - **knee point**: 76% of max for MySQL; 87% of max for Tomcat.

![Graphs showing CoV and Average for (a) MySQL and (b) Tomcat](image)
Controller

How many BEs can we co-locate?

- Slacklimit: the lower bound of slack for allowing the growth of BE jobs.

Co-locating decisions:

\[
\text{Slack} = \text{SLA} - \text{currentTL};
\]

\[
\begin{align*}
\text{if slack < 0 then} & \quad \text{StopBE();} \\
\text{else if } 0 < \text{slack} < \frac{\text{slackLimit}}{2} & \quad \text{CutBE();} \\
\text{else if } \frac{\text{slackLimit}}{2} < \text{slack} < \text{slackLimit} & \quad \text{DisallowBEGrowth();} \\
\text{else} & \quad \text{AllowBEGrowth();}
\end{align*}
\]

- contribution 1 < contribution 2
- slacklimit1 < slacklimit 2
Experimental Evaluation

- **Benchmarks:**
  - **LC services:**
    - Apache Solr: Solr engine+Zookeeper
    - Elasticsearch: Index+Kibana
    - Elgg: Webserver+Memcached+Mysql
    - Redis: Master + Slave
    - E-commerce: Haproxy+Tomcat+Amoeba+Mysql
  - **BE Tasks:**
    - CPU-Stress; Stream-LLC; Stream-DRAM
    - Iperf: Network
    - LSTM: Mixed
    - Wordcount
    - ImageClassify: deep learning

- **Testbed**
  - 16 Sockets, 64 GB of DRAM per socket. Each socket shares 20 MB of L3 cache.
  - Intel Xeon E7-4820 v4 @ 2.0 GHz: 32 KB L1-cache and 256 KB L2-cache per core.
  - The operating system is Ubuntu 14.04 with kernel version 4.4.0-31.
Overall analysis (compared to Heracles [ISCA,2015])

- Improve EMU (=LC throughput + BE throughput) by 11.6%~24.6%;
- Improve CPU utilization by 19.1%~35.3%;
- Improve memory bandwidth utilization by 16.8%~33.4%.
Timeline Analysis

Timeline:
- Time 3.3: `suspendBE();`
- Time 5.6: `allowBEGrowth();`
- Time 7.7: `cutBE();`
- Time 9.3: `suspendBE().`

Figure 17. The timeline of Rhythm’s running process.
Conclusion

- Rhythm, a deployment controller that maximizes the resource utilization while guaranteeing LC service`s tail latency requirement.
  - Request tracer
  - Contribution analyzer
  - Controller

- Experiments demonstrate the improvement on system throughput and resource utilization.
Thank you!

Questions?