13th Symposium on Software Performance

8th November 2022

Predicting Scaling Efficiency of Distributed Stream Processing Systems via Task Level Performance Simulation

Johannes Rank, Maximilian Barnert, Andreas Hein, Helmut Krcmar

Chair for Information Systems: Lab Krcmar
Technical University of Munich
johannes.rank@tum.de
Motivation

- Distributed stream processing systems are the backbone of many Big Data implementations and can reach a considerable size in terms of cores / workers
- **CPU efficiency** becomes *increasingly important* from both, an environmental as well as a cost perspective
- Most streaming systems allow for *flexibility regarding their scaling direction*
- Most DevOps do not know what scaling actually means in terms of CPU efficiency?
Example – Azure Hosting

Which Architecture would you **choose as a manager**?

- 2x Instance “A4 v2” (4 cores, 8GB RAM, 0.286$/h)  
  **Scale-Out**
  
  - 1x Instance “A8 v2” (8 cores, 16GB RAM, 0.600 $/h)

**Scale-Up**

- **Scale-Out architecture 4.66% cheaper**
Example – CPU Efficiency

Which Architecture would you **choose as a manager**?

- Example: Yahoo Streaming Benchmark with Apache Flink
- Workload: 600k events/s

➢ Scale-Up architecture **26.79%** more efficient
**Paper Topic**

**Question:** How efficient are 3, 4, 5 ... N workers?
Performing and comparing N measurements is not efficient

**Idea:** Performance Simulation of different cluster sizes (with PCM)

**Assumption:** We have a fixed number of cores and want to simulate how many workers we should distribute them to (e.g. 2x C6 or 1x C12)

**PCM Design Requirement:** Accurate approach that is quick&simple to implement
- No automation is in place that allows an easy PCM generation!
- One manually created PCM model that allows to predict different cluster sizes, without changing the model
  - No changes in the ResourceEnvironment, Allocation or System Model
  - Cluster size is specified as an input parameter of the Usage model
- Despite the quick&simple approach, the results should provide sufficient accuracy
PCM Design Requirement

Simulation Example:
- Workload = 600k events/s
- Workers = 4 (each 12vCPU)

12x IBM Power9 CPU cores (4.2 GHz)
Simultaneous Multithreading 4 (SMT4) ) 48 vCPU

quick&simple
Dynamic Resource Demands

- We know that the Resource Demands change in dependence of the number of workers.

- Usually we would need to model each cluster configuration as a separate combination Allocation+ResourceEnv+System Model.

- Instead we model the Resource Demand in dependence of the received events (the more events a node receives the more efficient it works).

  \[ 9.1259 \times \text{load\_per\_node} \times \text{VALUE} + 326.7 \times \text{CP} \ldots \]

- Therefore, we need a virtual load balancer that divides the total load through the number of workers.
Task-Level Performance Modelling

▪ The probably simplest approach would be to measure the total CPU utilization for a few cluster configurations and to perform a regression analysis.

![Graphs showing CPU utilization for different cluster configurations.](image)

▪ However, looking only at the total utilization is not accurate enough (abstraction level too high).
  – Each streaming task has its own efficiency curve that can either grow linear, logarithmic, polynomial or exponential to the workload.
  – The PCM Resource Effect Specification will model each task as an internal action with its own ResourceDemand.
Task-level Performance Modelling

How many events do we send AND receive during re-distribution?

Each worker **sends** \((\text{load\_per\_node}/\text{workers}) \times (\text{workers}-1)\) events to other workers and **receives** \((\text{load\_per\_node}/\text{worker}) \times (\text{worker}-1)\) from the other workers.

How to get the parametrization in dependence of the workers / load?
Task-level Measurement

Our toolchain proposed in (Rank, et al. 2020) profiles applications with BPF and combines the results with PMU measurements\(^1\)

- Identify Process Scope
  - Scope Analyzer.sh

- SPS Stack-trace sampling
  - stack_count.bt

- PMC Measurement
  - perf stat -p

- Workload Identification
  - workload_trace.bt

- Stackfold & FlameGraph
  - stack_folded.data

- Result Generation
  - results.sh

- Absolute Demand
  - Stream Engine: 133.5 bn Instr
  - App/Serial: 50.5 bn Instr
  - App/Join: 60 bn Instr

This way we get the consumed CPU cycles for each streaming task

---

Task-level Parametrization Approach

- Our approach requires to measure three cluster configurations
  - lowest (1 worker), highest (12 workers) and one in between (we chose)
  - $N=\text{workers}, \ C=\text{phys\_cores\_per\_worker}$ -> $N1\_C12, N2\_C6$ and $N12\_C1$

---

**VCPU**

---

### Step 1: Calculate CPU consumption per task and worker (divide by number of workers)

### Step 2: Calculate CPU cycles per event by multiplying core consumption with the processing rate (4.2 GHz) and dividing by the load (600k). The x-axis becomes the number of processed events on a single node for the respective cluster configuration (e.g. 600 for $N1\_C12$, 50 = $N12\_C1$). For KeyBy use the number of redistributed events instead.
Experiment

- “quick&simple”
  - PCM Model
  - Required model changes to simulate different cluster sizes
  - 3x Measurements for parametrization
  - Profiling approach (fully automated)

- “accurate”
  - Does the task-level prediction perform better?
  - **Baseline**: More accurate than a simple regression approach (that only looks at the total CPU consumption) based on the same number of measurements
  - Predict N3, N4, N5
Conclusion and Limitation

- Fast and easy PCM based prediction approach
- Achieves highly accurate results
- Can be applied to running systems (no instrumentation) required

- For the experiment we assumed a constant load (600k events/s). We did not test how accurate the prediction works for different load levels

- We only scaled our cluster from 1 to 12 worker nodes. We did not test how accurate the prediction works for even bigger cluster sizes
Thank you for your attention!

Questions?

mail: johannes.rank@tum.de
Azure Hosting

https://azure.microsoft.com/de-de/pricing/calculator/

(04.11.2022)
Azure Hosting

https://azure.microsoft.com/de-de/pricing/calculator/

(04.11.2022)