The HCl Scheduler

Going all-in on Heterogeneity

Michael Kaufmann, Kornilios Kourtis

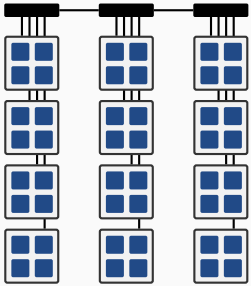
USENIX HotCloud'17

Karlsruhe Institute of Technology

Institute of Telematics

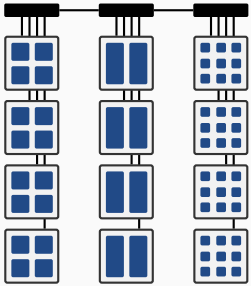
IBM Research - Zurich

Evolution of Clusters - Past



→ Mostly homogeneous hardware

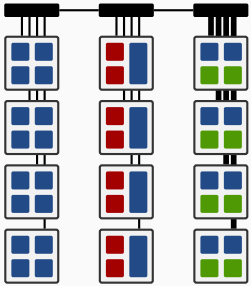
Evolution of Clusters



> Heterogeneity occurs because servers are gradually provisioned and replaced over the typical 15-year lifetime of a DC. At any point in time, a DC may host 3-5 server generations with a few hardware configurations per generation

C. Delimitrou and C. Kozyrakis. "Paragon: QoS-aware scheduling for heterogeneous datacenters." ACM SIGPLAN 2013.

Evolution of Clusters - Today & Future



IBM EXTENDS GPU CLOUD CAPABILITIES, TARGETS MACHINE LEARNING

The Next Platform, 2016-05-19

AWS announces FPGA instances for its EC2 cloud computing service

TechCrunch, 2016-11-30

Google launches GPU support for its Cloud Platform

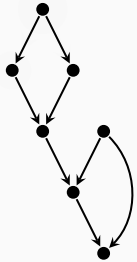
TechCrunch, 2017-02-21

Microsoft Azure ND-series offers more GPUs, power

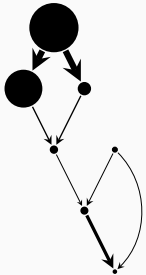
TechGenix, 2017-06-09

What about Applications?

Homogeneous View



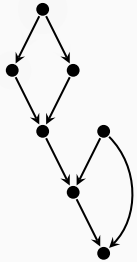
Heterogeneous View



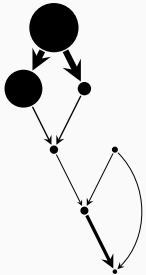
Spark stage graph of TPC-DS Query #44

What about Applications?

Homogeneous View



Heterogeneous View



Spark stage graph of TPC-DS Query #44

→ **Applications are heterogeneous as well**

- Runtime and resource requirements vary
- I/O volumes vary

Consequences

> Ignoring heterogeneity can lead to significant inefficiencies, as some workloads are sensitive to hardware configurations.
> *C. Delimitrou and C. Kozyrakis. "Paragon: QoS-aware scheduling for heterogeneous datacenters." ACM SIGPLAN 2013.*

→ We leave a lot of potential untapped, not least due to inefficient task scheduling.

The HCl Scheduler - Goals, Starting Point & Approach

➔ **Goals**

1. Efficient utilization & sharing of (heterogeneous) resources.
2. Minimize application (vs. task) runtime.
3. Reduce costs for clients and operators.

The HCl Scheduler - Goals, Starting Point & Approach

Goals

1. Efficient utilization & sharing of (heterogeneous) resources.
2. Minimize application (vs. task) runtime.
3. Reduce costs for clients and operators.

Starting Point

We primarily aim for maximizing scheduling quality and secondarily for maximizing scaling & throughput and minimizing latency.

The HCl Scheduler - Goals, Starting Point & Approach

Goals

1. Efficient utilization & sharing of (heterogeneous) resources.
2. Minimize application (vs. task) runtime.
3. Reduce costs for clients and operators.

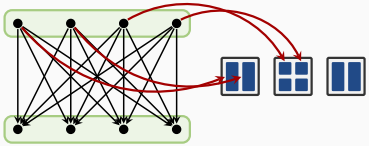
Starting Point

We primarily aim for maximizing scheduling quality and secondarily for maximizing scaling & throughput and minimizing latency.

Approach

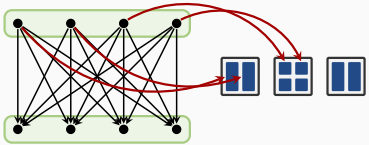
Exploit detailed resource and application information in order to find globally optimal application schedules.

Closer Look at the Problems

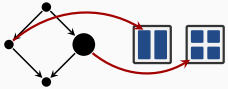


P1. Stragglers due to h/w selection

Closer Look at the Problems

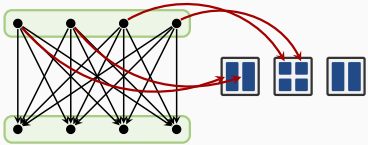


P1. Stragglers due to h/w selection

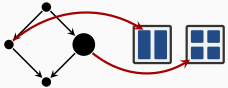


P2. Priority inversion

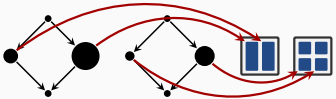
Closer Look at the Problems



P1. Stragglers due to h/w selection

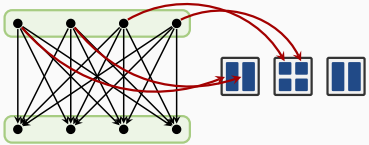


P2. Priority inversion

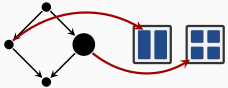


P3. Non-beneficial *stealing* of preferred resources

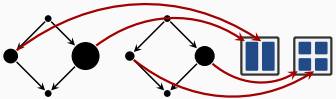
Closer Look at the Problems



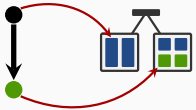
P1. Stragglers due to h/w selection



P2. Priority inversion

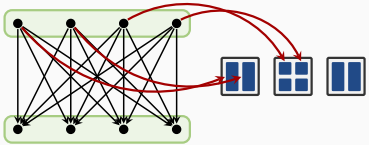


P3. Non-beneficial *stealing* of preferred resources

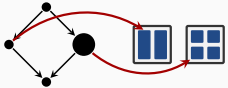


P4. Dominating I/O costs

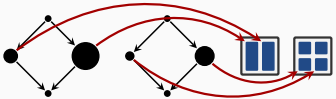
Closer Look at the Problems



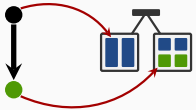
P1. Stragglers due to h/w selection



P2. Priority inversion



P3. Non-beneficial *stealing* of preferred resources



P4. Dominating I/O costs

The HCl Scheduler - Related Work

Scheduler	Year	H/W	App	Task	P1	P2	P3	P4
LHEFT	2010	(✔)	✔	✔	-	✔	-	(✔)
Mesos	2011	(✔)	-	-	-	-	-	-
Paragon	2013	✔	-	✔	-	-	-	-
Kubernetes	2014	(✔)	-	(✔)	-	-	-	-
Tetris	2014	-	(✔)	(✔)	(✔)	-	-	-
TetriSched	2016	(✔)	(-)	(✔)	(✔)	(✔)	-	-
Graphene	2016	(✔)	✔	(✔)	(✔)	(✔)	-	(✔)
Carbyne	2016	(✔)	✔	✔	(-)	-	(✔)	-
HCl	2017	✔	✔	✔	✔	✔	✔	✔

→ Heterogeneity of **applications and resources** is rarely a primary concern.

The HCl Scheduler - Related Work

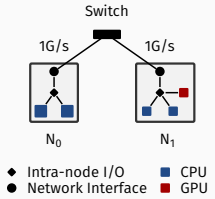
Scheduler	Year	H/W	App	Task	P1	P2	P3	P4
LHEFT	2010	(✔)	✔	✔	-	✔	-	(✔)
Mesos	2011	(✔)	-	-	-	-	-	-
Paragon	2013	✔	-	✔	-	-	-	-
Kubernetes	2014	(✔)	-	(✔)	-	-	-	-
Tetris	2014	-	(✔)	(✔)	(✔)	-	-	-
TetriSched	2016	(✔)	(-)	(✔)	(✔)	(✔)	-	-
Graphene	2016	(✔)	✔	(✔)	(✔)	(✔)	-	(✔)
Carbyne	2016	(✔)	✔	✔	(-)	-	(✔)	-
HCl	2017	✔	✔	✔	✔	✔	✔	✔

→ Heterogeneity of **applications and resources** is rarely a primary concern.

The HCl Scheduler - Overview

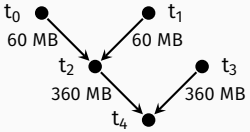
➔ HCl combines 3 sources of information that are usually separated.

Detailed Cluster Model



➔ Extracted from OS / Resource Manager

Annotated Application DAG



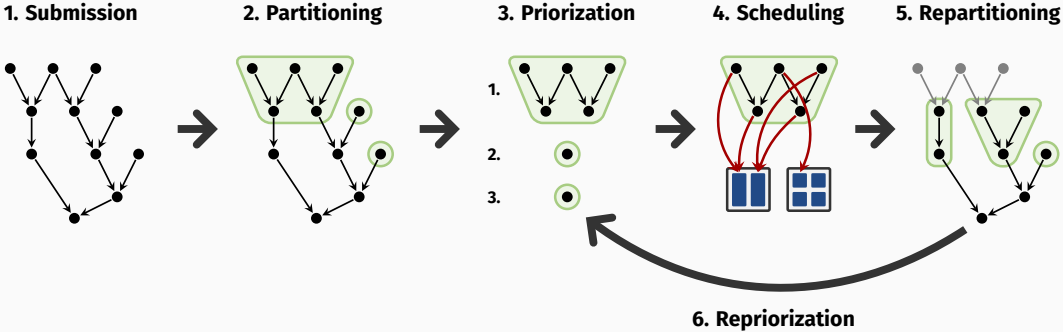
➔ Extracted from App Framework (e.g. Spark)

➔ Soft/hard constraints

Performance Database

Task	$N_{0_{cpu}}$	$N_{1_{cpu}}$	$N_{1_{gpu}}$
t_0	4.0s	5.0s	2.0s
t_1	4.0s	5.0s	2.0s
t_2	1.0s	1.5s	-
t_3	8.0s	10.0s	5.0s
t_4	1.0s	1.5s	-

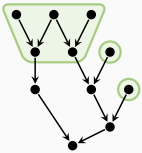
The HCl Scheduler - 10,000 Foot View



The HCl Scheduler - Heuristics

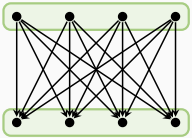
→ DAG scheduling is an NP-complete problem, hence heuristics are necessary to reduce the cost of scheduling to desired levels.

1. Partitions



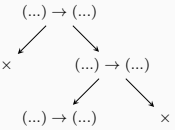
2. Node Classes

3. Task Classes



4. Sampling

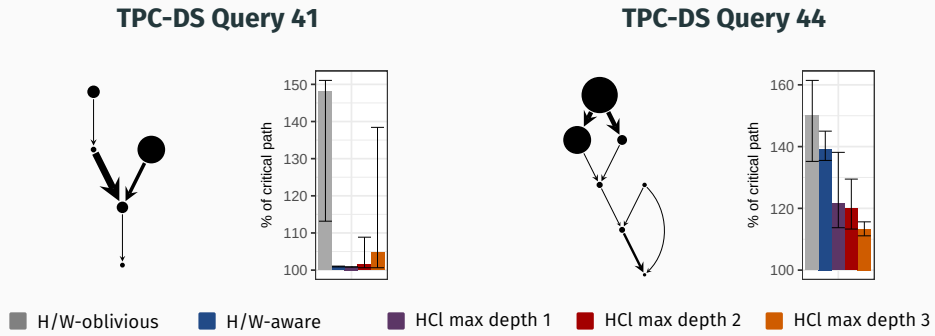
5. Early Termination*



* = Not a heuristic

Evaluation - Preliminary TPC-DS Benchmark Results (Excerpt)

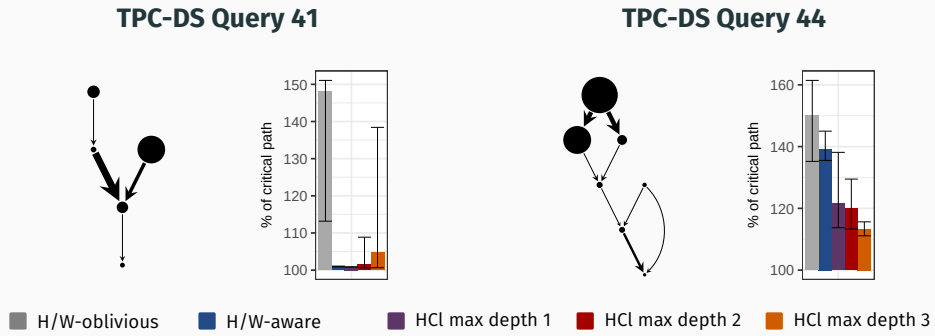
- Single-application, model-based evaluation (extracted from Spark) on an 8 node cluster (2 fast, 6 slow nodes, factor 1.5).



- HCl schedules are within **15%** of the critical path on average.
- HCl schedules are **48%** shorter on average.

Evaluation - Preliminary TPC-DS Benchmark Results (Excerpt)

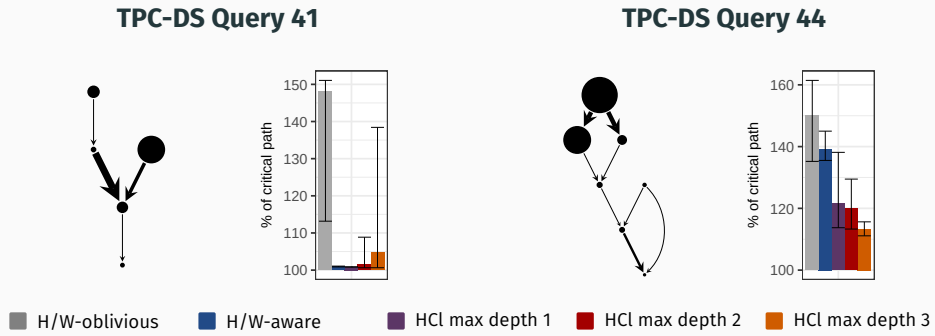
- Single-application, model-based evaluation (extracted from Spark) on an 8 node cluster (2 fast, 6 slow nodes, factor 1.5).



- HCl schedules are within **15%** of the critical path on average.
- HCl schedules are **48%** shorter on average.

Evaluation - Preliminary TPC-DS Benchmark Results (Excerpt)

- Single-application, model-based evaluation (extracted from Spark) on an 8 node cluster (2 fast, 6 slow nodes, factor 1.5).



- HCl schedules are within **15%** of the critical path on average.
- HCl schedules are **48%** shorter on average.

Conclusion

- Current cluster schedulers leave potential untapped.
  - We need to consider H/W and S/W heterogeneity in order to solve scheduling problems.
- The HCl Scheduler is our attempt at exploring the potential of a fully heterogeneity-aware scheduler.
  - Preliminary evaluation shows that we can significantly shorten schedules.
  - Few additionail faster resources can help to speed up applications significantly.
- Lots of work left to be done.

Outlook

Thank you! Questions?

Outlook

→ **Ongoing & Future Work**

- Improve performance (heuristics & implementation)
- Implement multiple resources support
- Integrate into distributed application framework (Apache Spark, TensorFlow)
- Evaluate opportunities at edge between scheduling, I/O (Crail Store integration) and H/W acceleration.

Backup

The HCl Scheduler - Example (Problem 3: Non-beneficial *stealing* of preferred resources)