Building Scalable and Flexible Cluster Managers Using Declarative Programming

Lalith Suresh, Joao Loff\textsuperscript{1}, Faria Kalim\textsuperscript{2}, Sangeetha Abdu Jyothis\textsuperscript{3}, Nina Narodytska, Leonid Ryzhyk, Sahan Gamage, Brian Oki, Pranshu Jain, Michael Gasch

VMware, \textsuperscript{1}IST (ULisboa) / INESC-ID, \textsuperscript{2}UIUC, \textsuperscript{3}UC Irvine and VMware
Hard to Build 😞

Cluster Managers

DCM 😊 Code-generate implementations from high-level specifications
Place us on the same rack!

Do **NOT** place us on the same rack!

POD
- 2GB RAM
- 16GB disk
- 1 core

Distribute us evenly!

**Kubernetes Scheduler**

30 hard and soft constraints

NP-Hard
Multi-dimensional bin-packing with constraints
Kubernetes Scheduler
Custom
Best-effort
Heuristics
Scalability?
Challenging with complex constraints

Decision quality?
Can miss feasible solutions

Extensibility?
Hard to add new policies and features

Custom
Best-effort
Heuristics
Our approach

Declarative Cluster Managers (DCM)

Cluster state
Database

Generated Code

Constraint Solver

DCM Compiler

Constraints In SQL

In SQL
Our approach

Declarative Cluster Managers (DCM)

Use cases

<table>
<thead>
<tr>
<th>Kubernetes Scheduler</th>
<th>VM Load Balancing Tool</th>
<th>Distributed Transactional Datastore</th>
</tr>
</thead>
</table>

- Scalability
- Decision quality
- Extensibility
Our approach

Declarative Cluster Managers (DCM)

Kubernetes Scheduler

Scalability
Up to 2x faster (p95) pod placement than Kubernetes Scheduler (500 node scale)

ECDF

End-to-end pod creation latency (s)
Our approach

Declarative Cluster Managers (DCM)

Use cases

<table>
<thead>
<tr>
<th>Kubernetes Scheduler</th>
<th>VM Load Balancing Tool</th>
<th>Distributed Transactional Datastore</th>
</tr>
</thead>
</table>

**Scalability**
Up to 2x faster (p95) pod placement than Kubernetes Scheduler (500 node scale)

**Decision quality**
4x better load balancing
2x faster pre-emption
Tightly constrained scenarios

**Extensibility**
Policies in <20 lines of SQL
Non-trivial features (Unified Pod/VM scheduling)
Programming Model

Cluster state
Database

Constraints
In SQL

Generated Code

DCM Compiler

Constraint Solver
Variable Columns

Pod | @Variable |
--- | ---

Node

Foreign Key
CREATE VIEW avoid_mem_overload AS

Select some rows

Predicate

Node Mem Overload
False
True
False

@hard constraint

Hard Constraints

@Variable

Pod | Node
---|---
? | ?
? | ?
? | ?

Constraint CHECK:

(nodes.mem_overload = false)
CREATE VIEW load_balance AS

Scalar expression to maximize

Soft Constraints

@ soft constraint

CREATE VIEW load_balance AS

Scalar expression to maximize
Programming Model

Express policies **concisely** using joins, aggregates, group bys, sub-queries, correlated sub-queries, arrays...
model = Model.create(dbConnection, constraints.sql);

model.solve();

Instantiate different models for different tasks and timescales
DCM Compiler

Cluster state
Database

Generated Code

Constraint Solver

DCM Compiler

Constraints
In SQL

This Photo by Unknown Author is licensed under CC BY 4.0
create view constraint_1 as
select * from t1 join t2 on t1.b = t2.b
where t2.e == 10
check(t1.c * t2.d = t2.c)

Flagship backend
Generates Java code that interfaces with
Google OR-Tools CP-SAT solver
```cpp
for (int t1_it = 0; t1_it < t1.size(); t1_it++) {
    for (int t2_it = 0; t2_it < t2.size(); t2_it++) {
        if (t1.get(t1_it).getB() == t2.get(t2_it).getB() && t2.get(t2_it).getE() == 10) {
            IntVar i1 = model.newIntVar(...);
            model.addProductEquality(i1, new IntVar[] {t1.get(t1_it).getCVar(), t2.get(t2_it).getDVar()});
            model.addEquality(t1_c[t1_it], i1);
        }
    }
}
```

**DCM Compiler**

- Iterate efficiently over tables
- Filter out rows
- Encode constraints

```sql
create view constraint_1 as
select * from t1 join t2 on t1.b = t2.b
where t2.e == 10
check(t1.c * t2.d = t2.c)
```

**Phase 1, SQL Parsing, program analysis**

**Phase 2, IR Convert to list comprehension IR, optimization passes**

**Phase 3, Backend Backend-specific optimization passes, generate code to efficiently traverse input tables and encode checks into low-level constraints**
Solver performance is highly sensitive to the encoding
Solver performance is highly sensitive to the encoding

\[ \begin{align*}
&\text{Constraint Propagator} \\
&[5, 6, 7] \\
&V_1 \neq V_2 \\
&[6, 8]
\end{align*} \]
Solver performance is highly sensitive to the encoding.

Solver fixes to 6 $[6] \quad v_1$
Solver performance is highly sensitive to the encoding

Constraint Propagator

$V_1 \neq V_2$

[6]  [6, 8]
Solver performance is highly sensitive to the encoding

Constraint Propagator

$V_1 \neq V_2$
Solver performance is highly sensitive to the encoding

- Reduce number of introduced variables and constraints
- Leverage specialized algorithms (i.e., *global constraints*)

Benchmark
Assign 50 tasks to 1000 workers
Naïve: 25 seconds
With optimizations: 85 ms!
# Evaluation

## Use cases

<table>
<thead>
<tr>
<th>Kubernetes Scheduler</th>
<th>VM Load Balancing Tool</th>
<th>Distributed Transactional Datastore</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Scalability</th>
<th>Decision quality</th>
<th>Extensibility</th>
</tr>
</thead>
</table>
Evaluation

- 500 node Kubernetes cluster
- Deploy a series of apps in an open-loop
- Azure 2019 trace
- Inter-pod anti-affinity constraint

Recommended best practice, but a challenging constraint
Kubernetes Scalability Evaluation

ECDF

End-to-end pod creation latency (s)
Kubernetes Scalability Evaluation

No anti-affinity constraints

50% of apps with anti-affinity constraints

100% of apps with anti-affinity constraints
Baseline samples only 50% of nodes

p95 latency

DCM = 5.33s
Baseline = 4.13s
Kubernetes Scalability Evaluation

ECDF

End-to-end pod creation latency (s)
DCM cuts 95\textsuperscript{th} percentile latency in half
More details in the paper!

Compiler internals, debugging, lessons learnt, DCM’s generality and limitations...

Building Scalable and Flexible Cluster Managers Using Declarative Programming

Lalith Suresh, João Loff¹, Faria Kalim², Sangeetha Abdu Jyothis, Nina Narodytska, Leonid Ryzhyk, Sahan Gamage, Brian Oki, Pranshu Jain, Michael Gasch

VMware, ¹IST (ULisboa) / INESC-ID, ²UIUC, ³UC Irvine and VMware

Abstract

Cluster managers like Kubernetes and OpenStack are notoriously hard to develop, given that they routinely grapple with hard combinatorial optimization problems like load balancing, placement, scheduling, and configuration. Today, cluster manager developers tackle these problems by developing systems that are often ad-hoc, inefficient, and maintenance-intensive.

Despite the complexity of the largely similar algorithmic problems involved, cluster managers in various contexts tackle the configuration problem using custom, systemspecific best-effort heuristics—an approach that often leads to a software engineering dead-end (§2). As new types of policies are introduced, developers are overwhelmed by having to write code to solve arbitrary combinations of increasingly...
Cluster state
Database
Constraints
In SQL

DCM
Code compiler

Generated Code

Constraint Solver

DCM +
Code-generate implementations from high-level specifications

Cluster Managers

Hard to Build 🥱

Kubernetes Scheduler
VM Load Balancing Tool
Distributed Transactional Datastore

Scalability
Decision quality
Extensibility

Thank you!
lsuresh@vmware.com
Code: https://github.com/vmware/declarative-cluster-management/