A flexible VM placement algorithm for IaaS clouds
VM scheduler

VM queue

model

decisions

actuators

monitoring data
SLOs and infrastructures evolve

Dynamic Power Management (DRS 3.1)

VM-VM affinity (DRS)

Dedicated instances (EC2)

VM-host affinity (DRS 4.1)

MaxVMsPerServer (DRS 5.1)

The constraint needed in 2014

2016

apr. 2011
dec. 2012
adapt the VM placement depending on pluggable expectations

network and memory-aware migration scheduler, VM-(VM|PM) affinities, resource matchmaking, node state manipulation, counter-based restrictions, energy efficiency, discrete or continuous restrictions
spread(VM[2..3]);
preserve(VM1,'cpu', 3);
offline(@N4);

0'00 to 0'02: relocate(VM2,N2)
0'00 to 0'04: relocate(VM6,N2)
0'02 to 0'05: relocate(VM4,N1)
0'04 to 0'08: shutdown(N4)
0'05 to 0'06: allocate(VM1, 'cpu', 3)

The reconfiguration plan

BtrPlace
CHOCO
An Open-Source java library for constraint programming

deterministic composition
high-level constraints

the right model for the right problem

\[ \mathcal{X} = \{ x_1, x_2, x_3 \} \]
\[ \mathcal{D}(x_i) = [0, 2], \forall x_i \in \mathcal{X} \]
\[ \mathcal{C} = \begin{cases} 
  c_1 : x_1 < x_2 \\
  c_2 : x_1 + x_2 \geq 2 \\
  c_3 : x_1 < x_3 
\end{cases} \]
BtrPlace CSP
models a reconfiguration plan
1 transition model per element
pre-defined variables as hooks

\[\begin{align*}
boot(v \in V) & \triangleq D(v) \in \mathbb{N} \\
st(v) & = [0, H - D(v)] \\
ed(v) & = st(v) + D(v) \\
d(v) & = ed(v) - st(v) \\
d(v) & = D(v) \\
ed(v) & < H \\
d(v) & < H \\
h(v) & \in \{0, \ldots, |N| - 1\}
\end{align*}\]

relocatable(v \in V) \triangleq \ldots
shutdown(v \in V) \triangleq \ldots
suspend(v \in V) \triangleq \ldots
spread(vs \subseteq vms) \triangleq \ldots
\quad i, j \in vs, i \neq j, host(i) \neq host(j)
Static analysis for performance

local search to focus on supposed mis-placed VMs
Static analysis for performance

self-partitioning to solve independent sub-problems in parallel
solving latency

- load spike
- hardware failure

Virtual machines (x 1,000) on 5,000 nodes
Extensibility in practice

looking for a better migration scheduler

network and workload blind
Extensibility in practice
looking for a better migration scheduler

network and workload aware
Extensibility in practice
solver-side

**Network Model**
- blocking heterogeneous network
- cumulative constraints; +/- 300 sloc.

**Migration Model**
- memory and network aware
- +/- 200 sloc.

**Constraints Model**
- restrict the migration models
- +/- 100 sloc.
model ± 96% accurate
speed up
Energy-aware scheduling

decommissioning 2x24 servers to 24 more powerfull
32.45 years/cpu of experiments

custom OS images
full cluster reservation
trafic shaping
g5k-subnet
storage reservation
power sensors
supporting docker from a VM to a container life-cycle

- running
  - boot
  - suspend
  - resume
  - migrate
- ready
  - halt
- sleeping

States: running, ready, sleeping
supporting docker

simplify the life-cycle using constraints or customized life-cycles ...

Diagram:
- Running
  - Boot
  - Halt
  - Suspend
  - Resume
- Ready
- Sleeping

Excluded:
- Migrate
supporting docker

... or overcome the limitations

- running
- ready
- sleeping
- boot
- halt
- suspend
- resume
- cold-migration
- re-instantation
adhoc VM schedulers do not fit evolving requirements
move to a flexible scheduler to be proactive

http://BtrPlace.org

production ready        live demo        stable user API        documented
tutorials        issue tracker        support        chat room