Efficient and Robust Allocation Algorithms in Clouds under Memory Constraints

Olivier Beaumont, Lionel Eyraud-Dubois, Paul Renaud-Goud

Inria & University of Bordeaux
Bordeaux, France

9th Scheduling for Large Scale Systems Workshop
### Typical Cloud Computing scenario

- A number of clients submit services to a provider
  - Think of services as commercial websites
- Services have hardware requirements (CPU, memory, I/O, ...)
- Clients express demands: for each service, enough CPU must be available
  - (serve enough requests per second)

### Optimization

- Several services can be allocated to the same Physical Machine (PM)
- Optimize energy and resource usage: consolidation
- Allocation of services onto PMs:
  - MultiDimensional Bin Packing Problem
Introducing machine failures

Data centers are large

- Failures will happen
- Our approach: over-provisioning – allocate additional capacity
- Clients express a **reliability requirement** expressed as a probability, or as a cost penalty

Assumptions

- Static setting over a given time period (between two migration phases):
  - Compute an allocation at the start of the period
  - During the time period, some machines fail
  - Services should still be running (enough instances) at the end
- Machines fail independently with probability $f$
Problem formulation

Notations

- Identical machines, with capacity $C$ and memory $M$
- $ns$ services: demands $d_i$, reliability requirement $r_i$, memory $m_i$

**Variables** $A_{i,j}$: CPU capacity for service $i$ on machine $j$

$$Alive_{cpu,i} = \sum_{j=1}^{m} is\_alive_j \times A_{i,j},$$

where $is\_alive_j = 0$ with prob. $f$

Formulation

$$\text{minimize } m \text{ s.t. } \begin{cases} \forall i, \mathbb{P}(Alive_{cpu,i} < d_i) < r_i \\
\forall j, \sum_{A_{i,j} > 0} m_i \leq M \\
\forall j, \sum_{i=1}^{ns} A_{i,j} \leq C \end{cases}$$
Two-step algorithm

1. Load Balancing
   Relaxed formulation to compute resource usage (CPU, memory) for each service so as to satisfy reliability constraints

2. Packing
   Use column generation techniques to obtain a feasible packing
First step: focus on reliability

**Approximation by normal distribution**

Given an allocation $A_{i,j}$, computing $\mathbb{P}(\text{Alive}_{cpu_i} < d_i)$ is \#P-complete. (as hard as counting the \# of solutions to a knapsack problem)
First step: focus on reliability

Approximation by normal distribution

Given an allocation $A_{i,j}$, computing $P(Alive_{cpu_i} < d_i)$ is \#P-complete. $P(Alive_{cpu_i} < d_i) < r_i$ is approx. by $\sum_{j=1}^{m} A_{i,j} - B_i \sqrt{\sum_{j=1}^{m} A_{i,j}^2} \geq K_i$
First step: focus on reliability

Approximation by normal distribution

Given an allocation $A_{i,j}$, computing $\Pr(\text{Alive}_{\text{cpu}}_i < d_i)$ is \#P-complete. $\Pr(\text{Alive}_{\text{cpu}}_i < d_i) < r_i$ is approx. by $\sum_{j=1}^{m} A_{i,j} - B_i \sqrt{\sum_{j=1}^{m} A_{i,j}^2} \geq K_i$.

Relaxed formulation (lower bound)

minimize $m$ s.t.

\[
\begin{align*}
\forall i, \sum_{j=1}^{m} A_{i,j} - B_i \sqrt{\sum_{j=1}^{m} A_{i,j}^2} & \geq K_i \\
\sum_{j} \sum_{A_{i,j}>0} m_i & \leq mM \\
\sum_{j} \sum_{i=1}^{ns} A_{i,j} & \leq mC
\end{align*}
\]
First step: focus on reliability

Approximation by normal distribution

Given an allocation $A_{i,j}$, computing $\mathbb{P}(\text{Alive}_{\text{cpu}}_i < d_i)$ is $\#P$-complete.

$\mathbb{P}(\text{Alive}_{\text{cpu}}_i < d_i) < r_i$ is approx. by $\sum_{j=1}^{m} A_{i,j} - B_i \sqrt{\sum_{j=1}^{m} A_{i,j}^2} \geq K_i$

Relaxed formulation (lower bound)

minimize $m$ s.t.

\[
\forall i, \sum_{j=1}^{m} A_{i,j} - B_i \sqrt{\sum_{j=1}^{m} A_{i,j}^2} \geq K_i
\]

\[
\sum_j A_{i,j} > 0 \quad m = m\mathbb{M}
\]

\[
\sum_j \sum_{i=1}^{n_s} A_{i,j} \leq m\mathbb{C}
\]

Homogeneous allocations are dominant ($A_{i,j} = A_i$ or 0)
First step: focus on reliability

Approximation by normal distribution

Given an allocation $A_{i,j}$, computing $\mathbb{P}(\text{Alive}_{cpu_i} < d_i)$ is #P-complete.

$\mathbb{P}(\text{Alive}_{cpu_i} < d_i) < r_i$ is approx. by $\sum_{j=1}^{m} A_{i,j} - B_i \sqrt{\sum_{j=1}^{m} A_{i,j}^2} \geq K_i$

Relaxed formulation (lower bound)

$$\begin{cases} \forall i, \sqrt{n_i} > B_i \\ \sum_i n_i m_i \leq mM \\ \sum_i \frac{K_i}{B_i \sqrt{n_i}} \leq mC \end{cases}$$

Can be solved for fractional $n_i$.

Homogeneous allocations are dominant ($A_{i,j} = A_i$ or 0)
Second step: packing

- Solve the homogeneous pb to get “ideal” allocations $n_i, A_i$
- “Pack” them onto individual machines

Remember the approximate formulation

\[
\begin{align*}
\text{minimize } m \text{ s.t. } & \\
\forall i, \sum_{j=1}^{m} A_{i,j} - B_i \sqrt{\sum_{j=1}^{m} A_{i,j}^2} & \geq K_i \\
\forall j, \sum_{i, j > 0} m_i & \leq M \\
\forall j, \sum_{i=1}^{n} A_{i,j} & \leq C
\end{align*}
\]

“Splitting” a service always improves the first inequality
Second step: packing

- Solve the homogeneous pb to get “ideal” allocations $n_i, A_i$
- “Pack” them onto individual machines

Remember the approximate formulation

\[
\begin{align*}
\text{minimize } m \text{ s.t. } & \\
& \forall i, \sum_{j=1}^{m} A_{i,j} - B_i \sqrt{\sum_{j=1}^{m} A_{i,j}^2} \geq K_i \\
& \forall j, \sum A_{i,j} > 0 \quad m_i \leq M \\
& \forall j, \sum_{i=1}^{n} A_{i,j} \leq C
\end{align*}
\]

“Splitting” a service always improves the first inequality

Splittable bin packing with memory constraints

- Allocate $n_i A_i$ total capacity to service $i$
- Chunk of at most $A_i$ on each machine
- Each chunk uses $m_i$ memory
- At most $M$ memory used and $C$ capacity per machine
LP Formulation with configurations

Valid configuration \( C_c \)
- \( \forall i \), fraction \( x_{i,c} \) of the maximum capacity \( A_i \) devoted to service \( S_i \)
- \( c \) is valid iff \( \sum i x_{i,c} \leq C \) and \( \sum x_{i,c} > 0 m_i \leq M \)
- **Variable** \( \lambda_c \): number of machines with this configuration
- **Almost full** configurations are enough:
  \[ x_{i,c} = 1 \text{ or } 0 \text{ except for one service} \]

Configuration formulation

\[
\text{minimize} \sum_{c \in \mathcal{F}} \lambda_c \quad \text{st} \quad \forall i, \sum_{c \in \mathcal{F}} \lambda_c x_{i,c} \geq n_i
\]
Column generation

**Primal:** minimize $\sum_{c \in F} \lambda_c$ st $\forall i, \sum_{c \in F} \lambda_c x_{i,c} \geq n_i$

**Dual:** maximize $\sum_i n_i p_i$ st $\forall c \in F, \sum_i x_{i,c} p_i \leq 1$

**Principle:** iteratively augment the set of configurations $F$

- Start with a small set of configurations
- Solve the primal, get a sub-optimal solution
- Find a violated constraint in the dual (splittable knapsack)
- Add this configuration to the set and loop
Splittable knapsack

Splittable knapsack problem

- Given $A_i, m_i, p_i, M$ and $C$,
- Find $J$ and $x_i$ such that $\sum_{i \in J} m_i \leq M$, and $\sum_{i \in J} x_i A_i \leq C$
- So as to maximize $\sum_{i \in J} x_i p_i$

Properties

- Almost full configurations are dominant.
- NP-hard (from 2-Part)
- $O(MC)$ Dynamic Programming algorithm to solve optimally.
Summary

Two-step heuristic

- Compute a homogeneous lower bound $n_i, A_i$ (binary search)
- Solve the packing problem
  - Start with a small set of configurations
  - Solve the primal, get a sub-optimal solution
  - Find a violated constraint in the dual (splittable knapsack)
  - Add this configuration to the set and loop
## Experimental evaluation

### Setting
- Memory and CPU usage from public Google trace: 150 jobs account for 90% of resource usage
- $r_i = 10^{-X}$, with $X \sim U(2, 8)$
- $f = 0.01$

### NoSharing heuristic
- Computes homogeneous allocations with $\forall i, A_i = C$
- Exactly one service per machine
Experimental results: number of machines

Running time of ColumnGeneration: mean 14.6s, max 22.2s
Experimental results: execution time
Conclusions

Explored allocation problems with reliability constraints
- Binomial approximation allows a good lower bound
- Reformulations & column generation give good heuristics

Further directions
- Extensions: multi-dimensional (CPU + IO), heterogeneous machines, ...
- Migrations for dynamic setting
- Recycle the techniques for other problems