Affinity-Aware Capacity Planning for Scheduling Long-Running Applications in Shared Clusters

Clément Mommessin

University of Leeds, UK

16 Feb. 2022
Large-Scale Shared Compute Clusters

1. **Long-Running Applications** (e.g., web services, DB, microservices)
   - Lasts several hours, up to months

2. **Short-lived batch jobs** (e.g., map-reduce, DAGs of tasks)
   - Lasts a few seconds, up to a few minutes
Capacity Planning

Long-Running Applications (LRAs)

- **Execution**: From time 0 to “infinity”
- **Resource requests**: Peak usage or time-varying profiles (CPU cores, memory, disk, GPU, ...)
- **Replicas**: Identical copies (for fault tolerance, service availability, ...)
- **Affinity constraints**: Limits replica co-location of different LRAs

Compute Cluster

- **Identical machines (or nodes)**
- **Resource capacity**: CPU cores, memory, disk, GPU, ...

→ Close to VM placement problems
Capacity Planning

Long-Running Applications (LRAs)

- **Execution**: From time 0 to “infinity”
- **Resource requests**: Peak usage or time-varying profiles (CPU cores, memory, disk, GPU, ...)
- **Replicas**: Identical copies (for fault tolerance, service availability, ...)
- **Affinity constraints**: Limits replica co-location of different LRAs

→ we do **not** consider self affinity constraints (can be easily included)
→ we do **not** consider affinity between nodes and LRAs
→ more intuitively “anti-affinity” constraints
## Capacity Planning

### Problem Formulation

**Inputs:**
- A large set of LRAs
- An “unlimited” set of nodes

**Objective:** Minimize the number of nodes used to accommodate all LRA replicas, under node capacity and LRA affinity constraints

**Target:** (very) large-scale scenarios, up to 10k – 100k nodes and LRAs

---

... later on:

1. Cluster consolidation, allocation of new LRAs
2. Dynamic scheduling of batch jobs

→ A story for the next time
# Problem Formulation

**Inputs:**
- A large set of LRAs
- An “unlimited” set of nodes

**Objective:** Minimize the number of nodes used to accommodate all LRA replicas, under node capacity and LRA affinity constraints

**Target:** (very) large-scale scenarios, up to 10k – 100k nodes and LRAs

→ **Not** really a scheduling problem: LRAs run infinitely

⇒ Generalization of Vector Bin Packing

   LRAs $\leftrightarrow$ items
   
   nodes $\leftrightarrow$ bins
**d-dimensional Vector Bin Packing** [Kou and Markowsky, 1977]

**Input:**
- A set of items $i$, with a size $s_{ih}$ in each dimension $h$
- An “unlimited” set of bins, with a capacity $C_h$ in each dimension $h$

**Objective:** Minimize the number of bins used to pack all items

Extensions for LRAs:
- Each item $i$ has a set of identical **replicas** $R_i$
- **Affinity constraints** represented as a directed “affinity graph”
Vector Bin Packing

→ The number of dimensions corresponds to the number of resource types
→ This is **not** Geometric Bin Packing nor “Alice déménage”

With peak usage resource requests \((d = 3)\):
Vector Bin Packing

→ The number of dimensions corresponds to the number of resource types
→ This is **not** Geometric Bin Packing nor “Alice déménage”

With peak usage resource requests \((d = 3)\):

With time-varying resource requests \((d = 3 \times T)\):
Trivial extension from 1-dimensional Bin Packing:

\[ LB = \max_{1 \leq h \leq d} \left\{ \left\lceil \sum_i \frac{|R_i|s_{ih}}{C_h} \right\rceil \right\} \]

→ There are more sophisticated LBs in 2 dimensions

[Caprara and Toth, 2001]

⇒ Open question: Can we include affinity constraints?
# 2-Dimensional Example

<table>
<thead>
<tr>
<th>LRA</th>
<th>replicas</th>
<th>cores</th>
<th>memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

Bin capacities: 8 / 16  
Lower bound: 3

**Affinity graph**
### 2-Dimensional Example

<table>
<thead>
<tr>
<th>LRA</th>
<th>replicas</th>
<th>cores</th>
<th>memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

**Bin capacities:** 8 / 16

**Lower bound:** 3

**Affinity graph**

M1 (5/4)  

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>
```
### 2-Dimensional Example

<table>
<thead>
<tr>
<th>LRA</th>
<th>replicas</th>
<th>cores</th>
<th>memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

**Bin capacities:** 8 / 16  
**Lower bound:** 3  

**Affinity graph**  

M1 (5/4)  

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A</th>
<th>A</th>
</tr>
</thead>
</table>

M2 (4/14)  

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>B</th>
</tr>
</thead>
</table>
2-Dimensional Example

<table>
<thead>
<tr>
<th>LRA</th>
<th>replicas</th>
<th>cores</th>
<th>memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

Bin capacities: 8 / 16
Lower bound: 3

Affinity graph

M1 (1/3)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

M2 (4/14)

<table>
<thead>
<tr>
<th></th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>
### 2-Dimensional Example

<table>
<thead>
<tr>
<th>LRA</th>
<th>replicas</th>
<th>cores</th>
<th>memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

**Bin capacities:** 8 / 16  
**Lower bound:** 3

**Affinity graph**

```mermaid
diagram affinity
    A -> B(2)
    B -> D(1)
    D -> E(0)
    A -> C(3)
    C -> E(2)
    B -> A(5)
```

<table>
<thead>
<tr>
<th>M1 (1/3)</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>A</td>
<td></td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M2 (4/14)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M3 (4/12)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>
2-Dimensional Example

<table>
<thead>
<tr>
<th>LRA</th>
<th>replicas</th>
<th>cores</th>
<th>memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

Bin capacities: 8 / 16
Lower bound: 3

Affinity graph

Affinity graph

M1 (1/3)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A</th>
<th>A</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>B</td>
<td></td>
<td></td>
<td>C</td>
</tr>
</tbody>
</table>

M2 (0/6)

|    | B | B |    | E  |

M3 (4/12)

| D  | D |    |    |

M4 (4/8)

| E  |    |    |    |
Our Solution

**General idea:** Adapt existing algorithms to handle replicas and affinity constraints

**2 main approaches**

1. **Application-centric Packing:** Classical Bin Packing algorithms (First Fit, Best Fit, Worst Fit) [Coffman et al., 1984]
   → For each LRA, allocate its replicas into current nodes; Activate a new node only when necessary

2. **Node-centric Packing:** Inspired by scheduling of LRAs [Panigrahy et al., 2011]
   → While there are unallocated LRAs: activate a new node and pack as much replicas as possible
Algorithm 1: Application-Centric Packing

1. Order LRAs following a specific policy
2. for each LRA do
   3. for each replica do
   4. Allocate the replica on the first node where it fits
   5. if no such node then
   6. Allocate the replica on a newly activated node
   7. Re-order the nodes following a specific policy

- Ordering policy for LRAs: arbitrary or Decreasing size
- Ordering policy for nodes: First Fit (arbitrary), Best Fit (increasing residual capacity), Worst Fit (decreasing residual capacity)
Algorithm 2: Node-Centric Packing

1. while there are unallocated replicas do
2. Activate a new node
3. while some replicas fit into that node do
4. Select the next LRA to allocate following a specific policy
5. Allocate replicas on the node

▶ LRA selection policy: “largest” that fits
Ordering Problem

In multiple dimensions, there is no strict ordering of a vector (of LRA sizes or node residual capacities)

Example: What is the largest item among \((1, 0), (0.4, 0.6), (0, 1)\)? How to order them?

Solution: Transform the vector to a scalar value

- **Application-centric packing**: introduce measures for LRA size or node residual capacity
- **Node-centric packing**: introduce scores for pairs of LRA size and node residual capacity
## LRA Size Measures

<table>
<thead>
<tr>
<th>Measure</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>( s_i = \sum_{h=1}^{d} s_{ih} )</td>
</tr>
<tr>
<td>Max</td>
<td>( s_i = \max_h { s_{ih} } )</td>
</tr>
<tr>
<td>Average Exponential</td>
<td>( s_i = \sum_{h=1}^{d} e^{\varepsilon A_h} \cdot s_{ih} ) [Panigrahy et al., 2011]</td>
</tr>
<tr>
<td>Surrogate</td>
<td>( s_i = \sum_{h=1}^{d} \lambda_h s_{ih} ) [Caprara and Toth, 2001]</td>
</tr>
<tr>
<td>Extended Sum</td>
<td>( s_i = \sum_{h=1}^{d} \frac{</td>
</tr>
</tbody>
</table>

with

- \( A_h \) — Average resource demand in dimension \( h \)
- \( \lambda_h \) — Total resource demand normalized in dimension \( h \)
- \( T_h \) — Total resource demand in dimension \( h \)
- \( |R_i| \) — Number of replicas of LRA \( i \)

→ Similar formulae for node residual capacity measures
### LRA - Node Scores

<table>
<thead>
<tr>
<th>Method</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>DotProduct</td>
<td>$\xi_{in} = \sum_{h=1}^{d} s_{ih} \overline{C}_{nh}$</td>
</tr>
<tr>
<td>[Panigrahy et al., 2011]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fitness</td>
<td>$\xi_{in} = \sum_{h=1}^{d} \frac{s_{ih}}{T_{h}} \cdot \frac{\overline{C}<em>{nh}}{\sum</em>{k \in \text{all nodes}} \overline{C}_{kh}}$</td>
</tr>
<tr>
<td>[Cai et al., 2021]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2Norm</td>
<td>$\xi_{in} = - \sum_{h=1}^{d} \left( \overline{C}<em>{nh} - s</em>{ih} \right)^2$</td>
</tr>
<tr>
<td>[Panigrahy et al., 2011]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>TightFill</td>
<td>$\xi_{in} = \sum_{h=1}^{d} \frac{s_{ih}}{\overline{C}_{nh}}$</td>
</tr>
</tbody>
</table>

with $\overline{C}_{nh}$ – Residual capacity of node $n$ in dimension $h$

$T_{h}$ – Total resource demand in dimension $h$

→ “Largest” LRA becomes LRA of largest score with the current node
Application-Centric Variant

Worst Fit Decreasing with Initial Nodes

Transform the algorithm to its *decision version*:

- **New input**: fixed set of $m$ activated nodes
- **Output**: feasible allocation using $m$ nodes, or failure

Finding appropriate $m$

- Binary search between LB and UB
- Iterative decrease from UB to LB with fixed step
  → Use solution of First Fit as an Upper Bound (UB)

**Aim?** Reduce the number of “active” affinity constraints in a node

**How?** Applying WFD, replicas are allocated one by one into different nodes

→ Only works with Worst Fit, does not change the behavior of First Fit or Best Fit
Experiments: Algorithms

Our Algorithms

- **Application-centric**: First Fit and all combinations of First/Best/Worst Fit Decreasing or Replica Spreading (SpreadWFD) with any combination of measure
- **Node-centric** (NCD) algorithm with any combination of score

Literature Baselines

From Medea [Garefalakis et al., 2018]

- **Tag Popularity** (Medea-TP): First Fit Decreasing with degree in affinity graph as size measure
- **Node Candidates** (Medea-NC): First Fit, selecting at each step LRA with smallest number of feasible nodes

From LRASched [Cai et al., 2021]

- **Fitness** (LRASched-Fitness): Node-centric algorithm with Fitness score
Experiments: Instances

Alibaba Tianchi dataset
- 9,338 LRAs, with replicas ranging in 1 – 618
- CPU core and memory request, time-varying profiles of 98 points
- Very few affinity constraints (graph density < 0.05%), affinity values ranging in 0 – 5

Instance Creation
Extension of original dataset
- Same LRA setting (replica numbers, core and memory peak requirement)
- Larger density of affinity graph (1%, 5%, 10%)
- Different graph classes (arbitrary, normal, threshold)
→ 10 instances per combination, 90 instances in total

Node capacity: 64 CPU cores and 128 units of memory
Metrics: Deviation from Lower Bound and running time
Results Summary: 2 Dimensions and Arbitrary Graph

- LRASched-Fitness
- Medea-TP
- SpreadWFD-Avg-BinSearch
- 12%
- 11%
- 30
- 10
- 20
- WFD-AvgExp
- 10%
- 40
- 4%
- 5%
- 700
- 4000

Family of Application-Centric Algorithms

- Medea-NC
- Computation Time (sec)
- Effectiveness (Deviation from LB)
More Experiments

Similar results observed with

- Larger graph density and time-varying resource requirement
- Larger submission scale (10k, 50, 100k LRAs) and peak requirement
- Larger submission scale and time-varying resource requirement

→ Code and omitted figures available on github.com/DSSGroup-Leeds/binpacking-expe-ICDCS22-paper
Conclusion 1

Cluster Capacity Planning

- Extension of Vector Bin Packing with replicas and affinity constraints
- Broad set of affinity-aware algorithms with various trade-offs between effectiveness and running time

⇒ Work submitted in the ICDCS conference, waiting for reviews

Possible extensions:

- Cluster consolidation (problem (1'))
- Scheduling of batch jobs (problem (2))
- Integration in Kubernetes, production clusters, ...
Going Further

General Vector Packing

- No replicas, no affinity constraints
- Focus on small-/medium-size instances (optimal still hard to find with $n < 50$)
- Consider a broader set of measures and scores, using weights for each dimension

Warning!

You are entering a WIP zone
Extending Measures

Weighted measure formulae

- **Average**: \( s_i = \sum_h w_h s_{ih} \)
- **Max**: \( s_i = \max_h \{ w_h s_{ih} \} \)

with different formulae for weight \( w_h \)

<table>
<thead>
<tr>
<th>Unit</th>
<th>( w_h = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>( w_h = T_h )</td>
</tr>
<tr>
<td>Exponential</td>
<td>( w_h = e^{\varepsilon D_h} )</td>
</tr>
<tr>
<td>Surrogate</td>
<td>( w_h = \frac{T_h}{\sum_{k=1}^d T_k} )</td>
</tr>
<tr>
<td>Inverse Average</td>
<td>( w_h = \frac{1}{T_h} )</td>
</tr>
</tbody>
</table>

... and their dynamic versions

- depending on remaining item sizes
- depending on bin residual capacities
Extending Scores

Weighted score formulae

- **DotProduct**: $\xi_{in} = a_{in} \sum_{h=1}^{d} s_{ih} \bar{C}_{nh}$
- **Tightfill**: $\xi_{in} = a_{in} \sum_{h=1}^{d} \frac{s_{ih}}{\bar{C}_{nh}}$
- **Fitness**: $\xi_{in} = a_{in} \sum_{h=1}^{d} \frac{s_{ih}}{T_{h}} \cdot \frac{\bar{C}_{nh}}{T'_{h}}$
- **L2Norm**: $\xi_{in} = -a_{in} \sum_{h=1}^{d} (\bar{C}_{nh} - s_{ih})^2$
- ... 

with different formulae for weight $a_{in}$ [Gabay and Zaourar, 2016]

<table>
<thead>
<tr>
<th>Unit</th>
<th>$a_{in}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit</td>
<td>$a_{in} = 1$</td>
</tr>
<tr>
<td>Inverse Product Norm2</td>
<td>$a_{in} = \frac{1}{|s_{i}|<em>2 \cdot |\bar{C}</em>{n}|_2}$</td>
</tr>
<tr>
<td>Inverse Bin Norm2 Squared</td>
<td>$a_{in} = \frac{1}{(|\bar{C}_{n}|_2)^2}$</td>
</tr>
</tbody>
</table>

→ Weights should depend (only?) on the dimension $h$
A New Approach

Scheduling-Based

**Setting:**
- Bins are machines
- Items are jobs
- Multiple dimensions can be viewed as multiple time dimensions

**Decision problem:**
- Does it exist a packing of items into at most \( m \) bins of capacities \( C_h \)?
- Does it exist a scheduling on \( m \) machines with makespans at most \( C_h \)?

\[ \Rightarrow \text{List Scheduling with LPT (Largest Processing Time) is equivalent to WFD with initial set of } m \text{ bins (} WFD(m) \text{)} \]

\[ \Rightarrow \text{Use binary search to find the smallest feasible } m \]

**but ... is it correct?**
Some Difficulties

Problems of monotonicity

- Scheduling anomalies [Graham, 1969]
- Anomalies of FFD with increased bin capacity [Coffman et al., 1978]

Item list (44, 24, 24, 22, 23, 17, 8, 8, 6, 6)

3 bins with $C = 60$

4 bins with $C = 61$

Open question: Is $WFD(m)$ monotone when $m$ increases?
Monotonicity of $WFD(m)$

Proof on monotonicity

Need to show for any $m' \leq m \leq m''$:

1. If $WFD(m)$ is feasible, then $WFD(m'')$ is feasible
2. If $WFD(m)$ is not feasible, then $WFD(m')$ is not feasible

Our current results:

- In dimension 1, $WFD(m)$ is monotone (and $WF(m)$ as well)
- With $d \geq 2$, we found anomalies for $WF(m)$
- With $d \geq 2$, we (probably) found (yesterday) anomalies for $WFD(m)$

$\Rightarrow$ That means we cannot **confidently** use binary search to find the smallest value of $m$
**Vector Bin Packing**

- Very large collection of algorithms (100+) for Vector Packing
- Experiments on a large set of instances
- Small theoretical results

→ We target a journal publication

Possible extensions:

- Use machine learning to tune weights and scores
- Classify a subset of “best-performing” algorithms on certain types of instances

**Open problems:**

- Can we find a (good) Lower Bound for the packing problem that takes into account affinity constraints?
- Can we prove monotonicity of Worst Fit with an initial set of bins?
Bibliography

1977  Multidimensional Bin Packing Algorithms, Kou and Markowsky
1978  An application of bin-packing to multiprocessor scheduling, Coffman, Garey and Johnson
1984  Approximation Algorithms for Bin Packing - An Updated Survey, Coffman, Garey and Johnson
2001  Lower bounds and algorithms for the 2-dimensional vector packing problem, Caprara and Toth
2011  Heuristics for Vector Bin Packing, Panigrahy, Talwar, Uyeda and Wieder
2016  Vector bin packing with heterogeneous bins: application to the machine reassignment problem, Gabay and Zaourar
2018  Medea: Scheduling of Long Running Applications in Shared Production Clusters, Garefalakis, Karanasos, Pietzuch, Suresh and Rao
Affinity-Aware Capacity Planning for Scheduling Long-Running Applications in Shared Clusters

Clément Mommessin

University of Leeds, UK

16 Feb. 2022