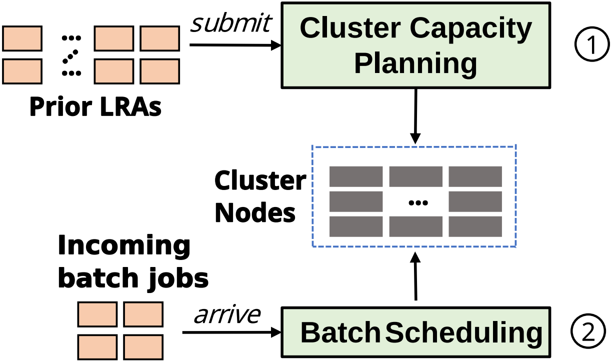
Affinity-Aware Capacity Planning for Scheduling Long-Running Aplications in Shared Clusters

Clément Mommessin

University of Leeds, UK

16 Feb. 2022

Large-Scale Shared Compute Clusters

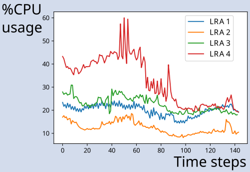


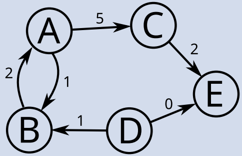
1. **Long-Running Applications** (e.g., web services, DB, microservices)
   → Lasts several hours, up to months
2. Short-lived batch jobs (e.g., map-reduce, DAGs of tasks)
   → Lasts a few seconds, up to a few minutes

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

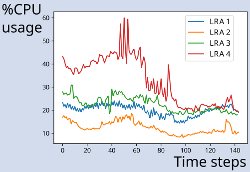
- **Identical machines (or nodes)**
- **Resource capacity**: CPU cores, memory, disk, GPU, . . .

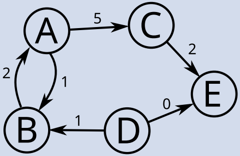
→ Close to VM placement problems

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 accomodate 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

$\rightarrow$ A story for the next time

Capacity Planning

Problem Formulation

Inputs:

- A large set of LRAs
- An "unlimited" set of nodes

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

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

$\rightarrow$ **Not** really a scheduling problem: LRAs run infinitely

$\Rightarrow$ Generalization of Vector Bin Packing

LRAs $\longleftrightarrow$ items
nodes $\longleftrightarrow$ bins

Vector Bin Packing

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

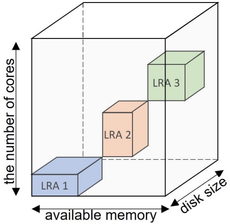
Extensions for LRAs:

- 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”

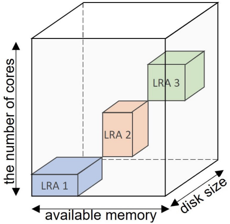
With peak usage resource requests ($d = 3$):



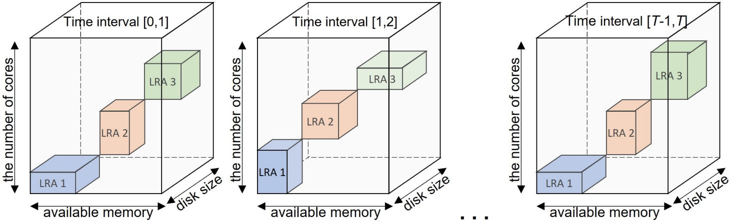
Vector Bin Packing

→ 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 ($d = 3 \times T$):



Lower Bound

Trivial extension from 1-dimensional Bin Packing:

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

$\rightarrow$ There are more sophisticated LBs in 2 dimensions [Caprara and Toth, 2001]

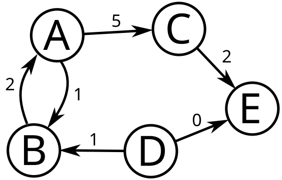
$\Rightarrow$ Open question: Can we include affinity constraints?

2-Dimensional Example

LRA	replicas	cores	memory
A	3	1	4
B	2	2	1
C	1	4	1
D	2	2	2
E	2	4	8

Bin capacities: 8 / 16

Lower bound: 3

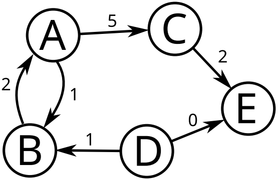


Affinity graph

2-Dimensional Example

LRA	replicas	cores	memory
A	3	1	4
B	2	2	1
C	1	4	1
D	2	2	2
E	2	4	8

Bin capacities: 8 / 16
Lower bound: 3



Affinity graph

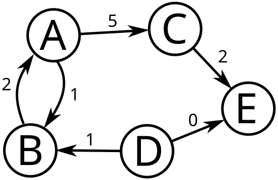
M1 (5/4)

2-Dimensional Example

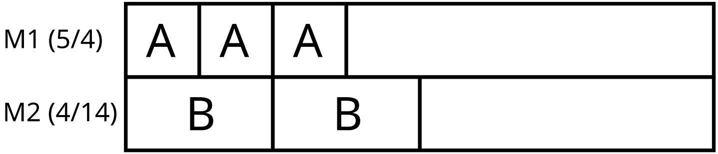
LRA	replicas	cores	memory
A	3	1	4
B	2	2	1
C	1	4	1
D	2	2	2
E	2	4	8

Bin capacities: 8 / 16

Lower bound: 3



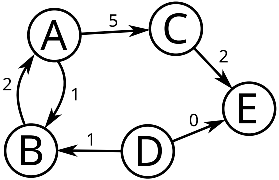
Affinity graph



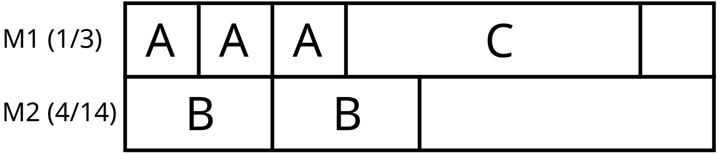
2-Dimensional Example

LRA	replicas	cores	memory
A	3	1	4
B	2	2	1
C	1	4	1
D	2	2	2
E	2	4	8

Bin capacities: 8 / 16
Lower bound: 3



Affinity graph

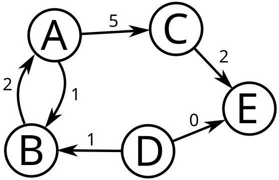


2-Dimensional Example

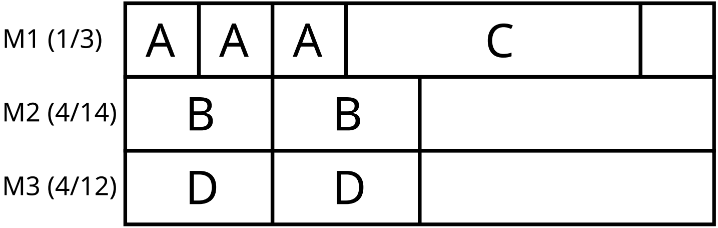
LRA	replicas	cores	memory
A	3	1	4
B	2	2	1
C	1	4	1
D	2	2	2
E	2	4	8

Bin capacities: 8 / 16

Lower bound: 3



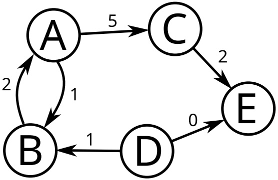
Affinity graph



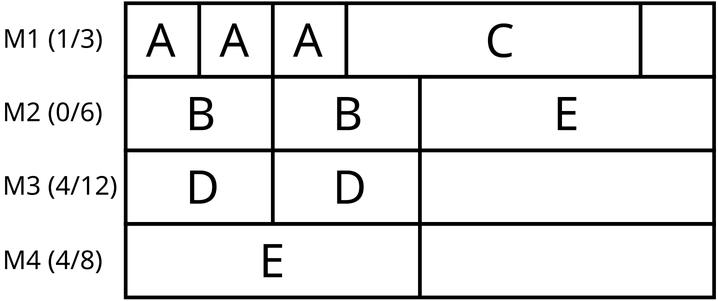
2-Dimensional Example

LRA	replicas	cores	memory
A	3	1	4
B	2	2	1
C	1	4	1
D	2	2	2
E	2	4	8

Bin capacities: 8 / 16
Lower bound: 3



Affinity graph



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]
   $\rightarrow$ 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]
   $\rightarrow$ While there are unallocated LRAs: activate a new node and pack as much replicas as possible

Algorithm Pseudocode

Algorithm 1: Application-Centric Packing

```
Order LRAs following a specific policy
for each LRA do
    for each replica do
        Allocate the replica on the first node where it fits
        if no such node then
            Allocate the replica on a newly activated node
        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 Pseudocode

Algorithm 2: Node-Centric Packing

```
while there are unallocated replicas do
    Activate a new node
    while some replicas fit into that node do
        Select the next LRA to allocate following a specific policy
        Allocate replicas on the node
```

- LRA selection policy: "largest" that fits

Ordering in Multiple Dimensions

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

Average	$s_i = \sum_{h=1}^{d} s_{ih}$
Max	$s_i = \max_h \{s_{ih}\}$
Average Exponential [Panigrahy et al., 2011]	$s_i = \sum_{h=1}^{d} e^{\varepsilon A_h} \cdot s_{ih}$
Surrogate [Caprara and Toth, 2001]	$s_i = \sum_{h=1}^{d} \lambda_h s_{ih}$
Extended Sum [Cai et al., 2021]	$s_i = \sum_{h=1}^{d} \frac{\lvert R_i \rvert}{T_h} s_{ih}$

with A_h – Average resource demand in dimension h

λ_h – Total resource demand normalized in dimension h

T_h – Total resource demand in dimension h

$|R_i|$ – Number of replicas of LRA i

$\rightarrow$ **Similar formulae for node residual capacity measures**

LRA - Node Scores

DotProduct [Panigrahy et al., 2011]	$\xi_{in} = \sum_{h=1}^{d} s_{ih} \overline{C}_{nh}$
Fitness [Cai et al., 2021]	$\xi_{in} = \sum_{h=1}^{d} \frac{s_{ih}}{T_h} \cdot \frac{\overline{C}_{nh}}{\sum_{k \in \text{all nodes}} \overline{C}_{kh}}$
L2Norm [Panigrahy et al., 2011]	$\xi_{in} = -\sum_{h=1}^{d} \left(\overline{C}_{nh} - s_{ih} \right)^2$
TightFill	$\xi_{in} = \sum_{h=1}^{d} \frac{s_{ih}}{\overline{C}_{nh}}$

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

$\rightarrow$ 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

$\rightarrow$ 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

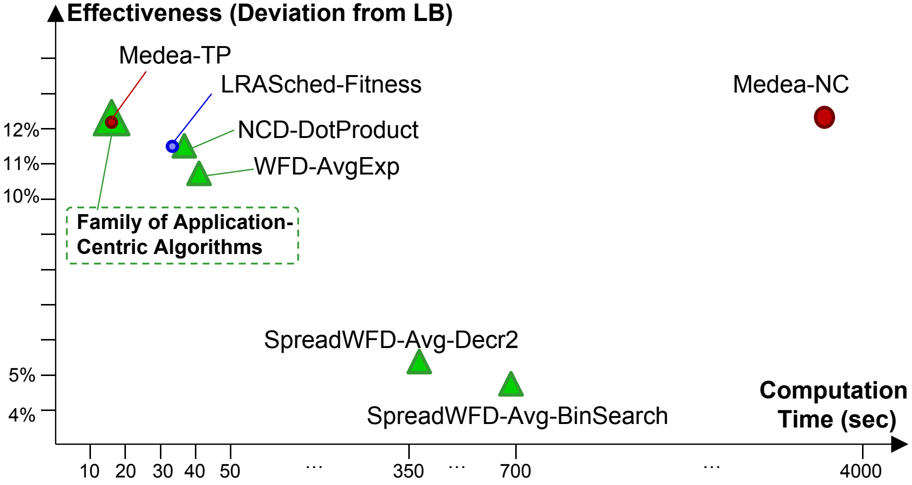
Extension of original dataset

- Same LRA setting (replica numbers, core and memory peak requirement)
- Larger density of affinity graph (1%, 5%, 10%)
- Different graph classes (*arbitrary*, *normal*, *threshold*)

$\rightarrow$ 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



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

Unit	$w_h = 1$
Average	$w_h = T_h$
Exponential	$w_h = e^{\varepsilon D_h}$
Surrogate	$w_h = \frac{T_h}{\sum_{k=1}^{d} T_k}$
Inverse Average	$w_h = \frac{1}{T_h}$

... and their dynamic versions

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

Extending Scores

Weighted score formulae

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

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

Unit	$a_{in} = 1$
Inverse Product Norm2	$a_{in} = \frac{1}{\|s_i\|_2 \cdot \|\overline{C}_n\|_2}$
Inverse Bin Norm2 Squared	$a_{in} = \frac{1}{(\|\overline{C}_n\|_2)^2}$

$\rightarrow$ 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?

$\Rightarrow$ List Scheduling with LPT (Largest Processing Time) is equivalent to WFD with initial set of m bins ($WFD(m)$)

$\rightarrow$ Use binary search to find the smallest feasible m

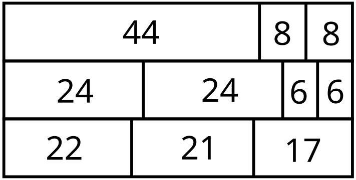
but . . . **is it correct?**

Some Difficulties

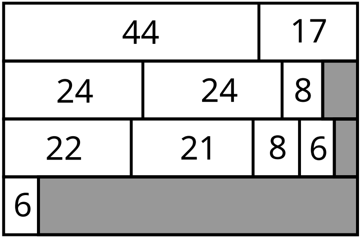
Problems of monotonicity

- 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

Conclusion 2

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 ceretain 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

2021 *LRASched: Admitting More Long-Running Applications via Auto-Estimating Container Size and Affinity*, Cai, Guo and Yu

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

Clément Mommessin

University of Leeds, UK

16 Feb. 2022