

# Allocation of Clients to Multiple Servers on Large Scale Heterogeneous Platforms

Olivier Beaumont, Lionel Eyraud-Dubois, Christopher Thaves-Caro

Laboratoire Bordelais de Recherche en Informatique  
Équipe CEPAGE (INRIA)

Scheduling in Knoxville  
14 May 2009

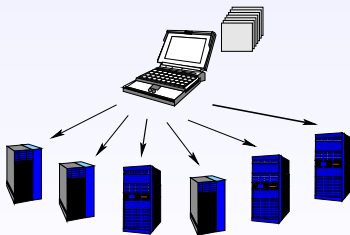
# Outline

- 1 Introduction
- 2 Independent tasks distribution
- 3 Online considerations
- 4 Conclusions

# Introduction

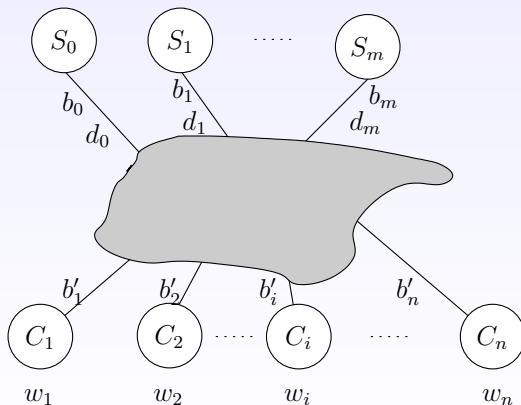
## Divisible Task Scheduling

- Master dispatches tasks to Workers
- Tasks can be arbitrarily divided
- Standard communication model: One Port



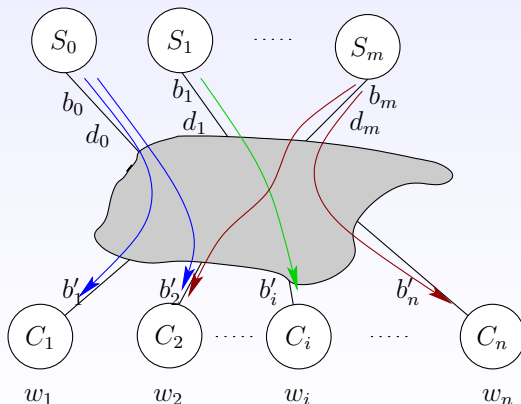
## Explore the Bounded Multi Port model

- Simultaneous communications, with a per-node bandwidth bound
- Internet-like: no contention inside the network
- Steady-state approach
- Keep things reasonable: degree constraint



## Explore the Bounded Multi Port model

- Simultaneous communications, with a per-node bandwidth bound
- Internet-like: no contention inside the network
- Steady-state approach
- Keep things reasonable: degree constraint



# Precise model

## An instance

- $m$  servers, with bandwidth  $b_i$  and maximal out-degree  $d_i$
- $n$  clients, with capacity  $w_j$

## A solution

- An assignment  $w_j^i$  of bandwidth from server  $i$  to client  $j$
- $\forall j, \sum_i w_i^j \leq b_j$  (capacity constraint at server  $j$ )
- $\forall j, \text{Card}\{i, w_j^i > 0\} \leq d_j$  (degree constraint at server  $j$ )
- $\forall i, \sum_j w_i^j \leq w_i$  (capacity constraint at client  $i$ )
- Maximise  $T = \sum_{i,j} w_j^i$

# Outline

- 1 Introduction
- 2 Independent tasks distribution
  - Complexity
  - Algorithm SEQ
  - Practical comparisons with heuristics
- 3 Online considerations
- 4 Conclusions

# Complexity

- NP-Hard: reduction from 3-Partition
  - ▶  $n$  servers with bandwidth  $B$  and degree 3
  - ▶  $3n$  clients with capacity  $a_i$ ,  $\sum a_i = nB$
  - ▶ Throughput  $nB$  reachable iff 3-Partition has a solution
- Easy to solve without the degree constraint
  - ▶ solve max-flow on the complete bipartite graph

→ Loosen the degree constraint



# Algorithm SEQ

- Resource augmentation: allowed one more connection per server
- Order clients by capacity
- For each server, bandwidth  $b$  and out-degree  $d$ :
  - ① Find a consecutive sublist of length  $d + 1$  such that:
    - ★ total capacity is at least  $b$
    - ★ capacity of the first  $d$  clients is less than  $b$
  - ② Assign these clients, perhaps split the last one
  - ③ Update the client list
- Choice of a subset does not matter
- Order of servers does not matter

# An example

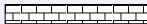
$$d_3 = 1$$

$$b_3 = 68$$



$$d_2 = 5$$

$$b_2 = 30$$



$$d_1 = 2$$

$$b_1 = 48$$



10 12

17

$w_4 = 24$

31

47

# An example

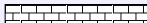
$$d_3 = 1$$

$$b_3 = 68$$



$$d_2 = 5$$

$$b_2 = 30$$



$$d_1 = 2$$

$$b_1 = 48$$



10 12

17

$w_4 = 24$

31

47

# An example

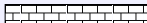
$$d_3 = 1$$

$$b_3 = 68$$



$$d_2 = 5$$

$$b_2 = 30$$



$$d_1 = 2$$

$$b_1 = 48$$



10 12

17

$w_4 = 24$

31

47

## An example

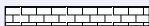
$$d_3 = 1$$

$$b_3 = 68$$



$$d_2 = 5$$

$$b_2 = 30$$



$$d_1 = 2$$

$$C(3, 5) = 72$$

$$b_1 = 48$$

$$C(3, 4) = 41$$



## An example

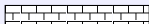
$$d_3 = 1$$

$$b_3 = 68$$



$$d_2 = 5$$

$$b_2 = 30$$



10

12

24

47



$$d_1 = 2$$

$$b_1 = 48$$

# An example

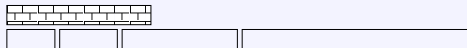
$$d_3 = 1$$

$$b_3 = 68$$



$$d_2 = 5$$

$$b_2 = 30$$



10

12

24

47



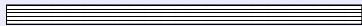
$$d_1 = 2$$

$$b_1 = 48$$

## An example

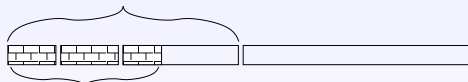
$$d_3 = 1$$

$$b_3 = 68$$



$$d_2 = 5$$

$$C(1, 3) = 46$$



$$b_2 = 30$$



$$d_1 = 2$$

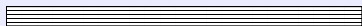
$$b_1 = 48$$



# An example

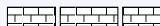
$$d_3 = 1$$

$$b_3 = 68$$



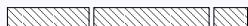
16

47



$$d_2 = 5$$

$$b_2 = 30$$



$$d_1 = 2$$

$$b_1 = 48$$

# An example

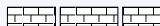
$$d_3 = 1$$

$$b_3 = 68$$



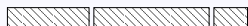
16

47



$$d_2 = 5$$

$$b_2 = 30$$



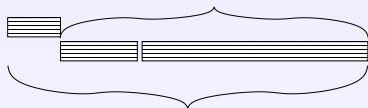
$$d_1 = 2$$

$$b_1 = 48$$

# An example

$$d_3 = 1$$

$$C(1, 2) = 63$$

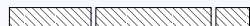


$$b_3 = 68$$



$$d_2 = 5$$

$$b_2 = 30$$

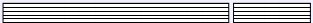


$$d_1 = 2$$

$$b_1 = 48$$

# An example

 5      Remaining server

  $d_3 = 1$   
 $b_3 = 63$

  $d_2 = 5$   
 $b_2 = 30$

  $d_1 = 2$   
 $b_1 = 48$

# Why does it work ?

Intuitively, more disparate client lists are “easier” to allocate

## Central Lemma

Define  $\mathcal{C} \preceq \mathcal{D}$  iff  $\forall k, \sum_{i=1}^k C_i \leq \sum_{i=1}^k D_i$

Statement: if  $\begin{array}{ccc} \mathcal{C} & \xrightarrow{\text{SEQ}(d+1,b)} & \mathcal{C}' \\ \preceq & & \\ \mathcal{D} & \xrightarrow{\text{valid}(d,b)} & \mathcal{D}' \end{array}$  then  $\mathcal{C}' \preceq \mathcal{D}'$

- Recursively,  $\mathcal{C}^{(m)} \preceq \mathcal{D}^{(m)}$ , thus  $\sum C_i^{(m)} \leq \sum D_i^{(m)}$
- Remaining client capacity is lower with SEQ than with any *valid* allocation

# Remarks

## Valid approximation algorithm

- At the end, remove the smallest client at each server
- $\forall j, T'_j \geq \frac{d_j}{d_{j+1}} T_j$
- $T' \geq \frac{d_{\min}}{d_{\min}+1} T \geq \frac{d_{\min}}{d_{\min}+1} T^*$

## Dual problem

- Given a throughput  $K$ , minimise the maximal degree  $d^*$  needed to reach  $K$
- SEQ with dichotomy achieves  $d^* + 1$

# Simple heuristics

## Largest Client Largest Server

Order clients and servers by capacities, and assign the currently largest client to the currently largest server. Split and reinsert the client if necessary.

## Largest Client Best Connection

Same as before, but sort servers by  $\frac{b_j}{d_j}$  (average available bandwidth).

## Online Best Connection

Same as LCBC, but without sorting clients first. Use the server with the closest average available bandwidth to the considered client

# Experimental setting

## Random instance generation

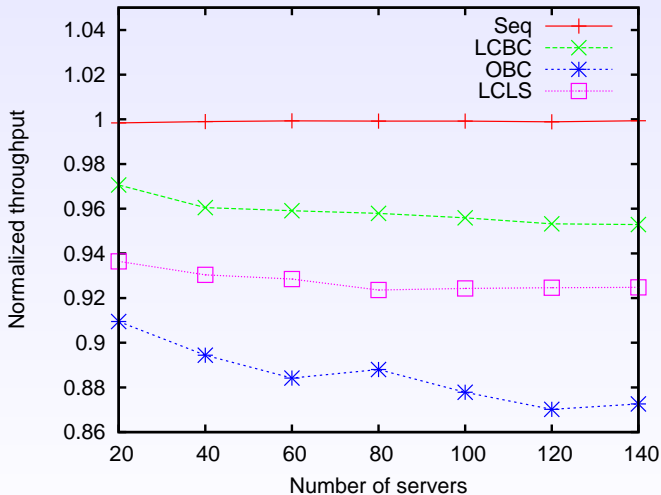
- $m$  servers,  $10m$  clients
- Capacities generated with power law distributions
- Server degrees nearly proportional to capacities

## Natural upper bounds

- $T \leq \sum_j b_j$
- $T \leq \sum_i w_i$
- Instances scaled so that both are roughly equal

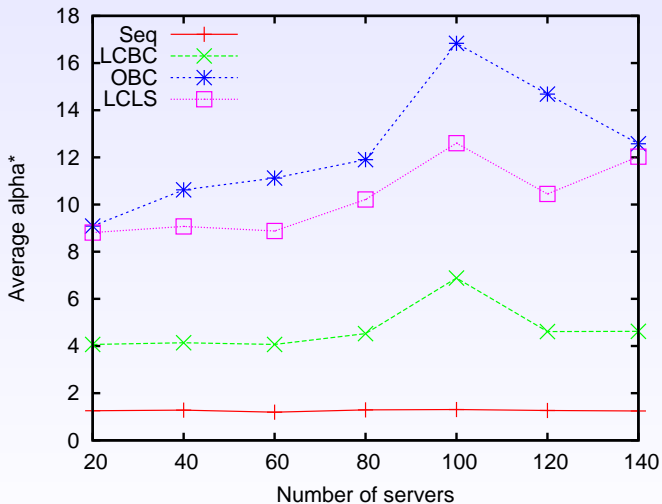


# Results



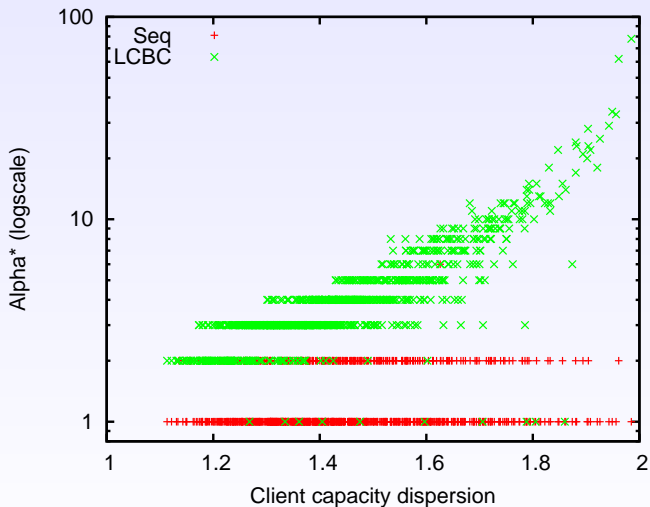
Average normalized throughput (over 250 instances) when  $m$  varies

# Results



Average  $\alpha^*$  when  $m$  varies

## Results



$\alpha^*$  values against dispersion for  $m = 80$

# Outline

- 1 Introduction
- 2 Independent tasks distribution
- 3 Online considerations**
- 4 Conclusions

# Can we do it online ?

## When clients come and go

- Disallow any change in the previous choices
- Count the number of changes for various algorithms

## In this section

- Fully online is impossible
- Online  $\text{SEQ}$  achieves a low number of changes

# Fully online is impossible

There is no fully online algorithm with resource augmentation factor  $\alpha$  and approximation ratio  $\frac{1}{k}$ .

- 1 server with bandwidth  $b = k \times 2^{\alpha k + 1}$  and degree  $k$
- $\alpha k$  groups of clients, group  $i$  having capacity  $2^i$
- one client of capacity  $b$
- $\mathcal{A}$  must connect at least one client from each group.
- $\Rightarrow$  No more connection available for the last client.

# Online SEQ

## Add some “locality” in SEQ

- Always choose the “rightmost” sublist of clients
- $\Rightarrow$  Ensures that the splitted client is reinserted at the same place

## Local transformations of client lists

- $\mathcal{C}$  is *increased* to  $\mathcal{C}^+$  by
  - ▶ insertion of a new client at position  $p$
  - ▶ capacity increase of  $\mathcal{C}_{p+1}$
- Similarly,  $\mathcal{C}$  is *decreased* to  $\mathcal{C}^-$  by
  - ▶ deletion of a client at position  $p$
  - ▶ capacity decrease of  $\mathcal{C}_{p+1}$

## Online SEQ

## Lemma

$$\text{If } \begin{array}{ccc} \mathcal{C} & \xrightarrow{\text{SEQ}(d+1,b)} & \mathcal{C}' \\ \downarrow & & \\ \mathcal{C}^+ = \mathcal{D} & \xrightarrow{\text{SEQ}(d+1,b)} & \mathcal{D}' \end{array} \quad \text{then } \mathcal{C}' \xrightarrow{+} \mathcal{D}'$$

Furthermore, the allocations differ by at most 4 changes.

Recursively, for a given set of servers  $\mathcal{S}$ ,  $\text{SEQ}(\mathcal{C} \cup \mathcal{C}_{\text{new}})$  and  $\text{SEQ}(\mathcal{C})$  differ by at most 4 changes per server.



# A comparison

## Aggressive Best Connection

- On client arrival, connect with Best Connection. If no room, remove the client that yields the largest gain.
- On client departure, use the newly available bandwidth to reduce the indegree of other clients. If there are unconnected clients left, act like on client arrival.

## On 80-server instances, with 500 events

- On average, throughput lower by 6%, can be as low as 75%
- Maximal number of changes for one event can reach 130 for one server
- Average number of maximal changes is 3.5 for SEQ, 1.6 for ABC

# Outline

- 1 Introduction
- 2 Independent tasks distribution
- 3 Online considerations
- 4 Conclusions**

## Summary

- Divisible Tasks, Multi-Port version
- Propose  $\text{SEQ}$ , a guaranteed approximation algorithm
- Analysed an online setting

## Future Works

- Broadcast – Streaming problem in the same model
- “P2P” setting: allow clients to forward messages
- Online algorithm with fewer total number of changes