Content

1. Architecture
2. Model
3. Algorithm
4. Develop
Content

1. Architecture
2. Model
3. Algorithm
4. Develop
Cross-VM Interference

<table>
<thead>
<tr>
<th></th>
<th>ArA</th>
<th>ArA-Global</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>1</td>
<td>Multiple</td>
</tr>
<tr>
<td>Service Rate</td>
<td>Homogenous</td>
<td>Heterogeneous</td>
</tr>
</tbody>
</table>
Architecture

Cross-VM Interference

<table>
<thead>
<tr>
<th>User</th>
<th>ServiceRate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Homogenous</td>
</tr>
<tr>
<td>Multiple</td>
<td>Heterogeneous</td>
</tr>
</tbody>
</table>
Content

1. Architecture
2. Model
3. Algorithm
4. Develop
Model ArA (Part 1)

Homogenous Service Rate

Arriving Tasks $\lambda$

Front-End Dispatcher

$\mu$

VM$_1$

Qlen=2 TaskUnits

VM$_2$

Qlen=4 TaskUnits

VM$_3$

Qlen=3 TaskUnits

Top-K

Task Amount As Queue Length

$\lambda$ uniform $(1,k)$
Model ArA (Part 1)

Homogenous Service Rate

Arriving Tasks $\lambda$

Front-End Dispatcher

uniform $(1,k)$

Shortest Waiting/Finish Time

Task Amount As Queue Length

Meaning of Queue Length

Same Service Rate

$\mu$

$\mu$

$\mu$

Top-K

Qlen=2 TaskUnits

Qlen=4 TaskUnits

Qlen=3 TaskUnits

Meaning of Queue Length
Online adjust $k$ candidates

Busy phase $\rightarrow$ large $k$ ($thl$)

Idle phase $\rightarrow$ small $k$ ($ths$)

Model ArA (Part2)
Model ArA-Global

(1) Heterogeneous Service Rate

Arriving Tasks $\lambda$ to Front-End Dispatcher

uniform $(1,k)$

Amount of Tasks

Expected Finish Time As Queue Length

$$QueueLength = ExpectedFinishTime = \frac{BufferTaskAmount}{ServiceRate}$$
Model ArA-Global

(1) Heterogeneous Service Rate

Arriving Tasks $\lambda$

Front-End Dispatcher

uniform $(1,k)$

Amount of Tasks

$Qlen=2$ TimeSlots

$Qlen=4$ TimeSlots

$Qlen=3$ TimeSlots

Expected Finish Time As Queue Length

$$QueueLength=ExpectedFinishTime=\frac{BufferdTaskAmount}{ServiceRate}$$
Model ArA-Global

ArA-Global

(1) Heterogeneous Service Rate

Arriving Tasks \( \lambda \) → Front-End Dispatcher → VM

- \( \text{Qlen}=2 \) TimeSlots
- \( \text{Qlen}=4 \) TimeSlots
- \( \text{Qlen}=3 \) TimeSlots

Amount of Tasks

\[ Q\text{Length}=\frac{\text{BufferedTaskAmount}}{\text{ServiceRate}} \]
Model ArA-Global

(2) Periodically Update Service Rate of Each VM

Arriving Tasks \( \lambda \)

Front-End Dispatcher

\( \text{uniform} \ (1,k) \)

Amount of Tasks

\[ \mu_1' = 0.5 \mu_1 \]

\( \text{Qlen=4 TimeSlots} \)

\( \mu_2' = 1.33 \mu_2 \)

\( \text{Qlen=4 TimeSlots} \)

\( \mu_3' = 3 \mu_3 \)

\( \text{Qlen=3 TimeSlots} \)

Update Queue Length
Model ArA-Global

(2) Periodically Update Service Rate of Each VM

Arriving Tasks $\lambda$

Front-End Dispatcher

$\text{uniform (1, k)}$

Amount of Tasks

$Q_{\text{len}=4 \text{ TimeSlots}}$

$Q_{\text{len}=4 \text{ TimeSlots}}$

$Q_{\text{len}=3 \text{ TimeSlots}}$

Update Queue Length

$\mu_1' = 0.5 \mu_1$

$\mu_2' = 1.33 \mu_2$

$\mu_3' = 3 \mu_3$
(2) Periodically Update Service Rate of Each VM

- Front-End Dispatcher
- Arriving Tasks: $\lambda$
- Amount of Tasks: Uniform $(1,k)$

VM 1: $\mu_1' = 0.5\mu_1$
VM 2: $\mu_2' = 1.33\mu_2$
VM 3: $\mu_3' = 3\mu_3$

Update Queue Length

- Expected Finish Time as Queue Length
- Amount of Tasks: $Q_{len}=2$ TimeSlots
- $Q_{len}=4$ TimeSlots
- $Q_{len}=3$ TimeSlots

Model ArA-Global
Model ArA-Global

(2) Periodically Update Service Rate of Each VM

Arriving Tasks \( \lambda \) → Front-End Dispatcher → Amount of Tasks

VM1: \( \mu_1' = 0.5 \mu_1 \)
VM2: \( \mu_2' = 1.33 \mu_2 \)
VM3: \( \mu_3' = 3 \mu_3 \)

Waiting Time is Not Fair

Update Queue Length

(1) Heterogeneous Service Rate

Front-End Dispatcher

Arriving Tasks

Amount of Tasks

Queue Length = Expected Finish Time

Buffered Task Amount

Service Rate

Expected Finish Time As Queue Length

Amount of Tasks

Qlen=2 TimeSlots
Qlen=4 TimeSlots
Qlen=3 TimeSlots

Waiting Time is Not Fair
Model ArA-Global

(3) Make Decision On Top-K Candidates Based On Expected Queue Length

Arriving Tasks $\lambda$

Front-End Dispatcher

Amount of Tasks $\text{uniform } (1,k)$

VM 1
$\mu_1' = 0.5\mu_1$

VM 2
$\mu_2' = 1.33\mu_2$

VM 3
$\mu_3' = 3\mu_3$

Expected Queue Length

- ExpectedQlen=6 TimeSlots
- ExpectedQlen=5 TimeSlots
- ExpectedQlen=7 TimeSlots

Top-K

Finish Time

Update Queue Length
(3) Make Decision On Top-K Candidates Based On Expected Queue Length

Arriving Tasks \( \lambda \) → Front-End Dispatcher → Top-K

- **VM1**: \( \mu_1' = 0.5\mu_1 \)
  - Expected Qlen = 6 TimeSlots

- **VM2**: \( \mu_2' = 1.33\mu_2 \)
  - Expected Qlen = 5 TimeSlots

- **VM3**: \( \mu_3' = 3\mu_3 \)
  - Expected Qlen = 7 TimeSlots

(2) Periodically Update Service Rate of Each VM

Arriving Tasks \( \lambda \) → Front-End Dispatcher → VMs

- VM1
  - Qlen = 4 TimeSlots
  - \( \mu_1' = 0.5\mu_1 \)

- VM2
  - Qlen = 4 TimeSlots
  - \( \mu_2' = 1.33\mu_2 \)

- VM3
  - Qlen = 3 TimeSlots
  - \( \mu_3' = 3\mu_3 \)

Update Queue Length

Finish Time
Content

1. Architecture
2. Model
3. Algorithm
4. Develop
Algorithm 1 The high level idea of the ARA-GLOBAL.

1: procedure ARA-GLOBAL
2:    /* 1. Initialize */
3: a. the large threshold thl for $K$; /* e.g., $thl = [0.5 \times N]$ */
4: b. the small threshold ths for $K$; /* e.g., $ths = 1$ */
5: c. information query delay: $D = d$
6: /* 2. VM Info Updating */
7: for each window of $D$ time do
8:    send queries to all computing sites for load information
9:    update load $Qlen$ and performance $ServiceRates$ info received from all VMs
10: end for each
11: /* 3. Detection of Changes in User Demands */
12: if Detect the start of burst then
13:    Increase $K$ to $thl$
14: else if Detect the start of idle then
15:    Decrease $K$ to $ths$
16: else
17:    Use $K$ for the site selection process
18: end if
19: /* 4. Update ExpectedFinishTime for each VM*/
20: for each VMs do
21:    \[\text{ExpectedFinishTime} = (\text{NewJobSize} + Qlen \times \text{OldServiceRate}) / \text{ServiceRate}\]
22: end for each
23: /* 5. Select Among Top-$K$ Shortest ExpectedFinishTime List*/
24: Sort all VMs $V_i$, $1 \leq i \leq N$, by ExpectedFinishTime List
25: set $V = V_1, V_2, ..., S_K$ /*Get $K$ sites with least load*/
26: set $s = \text{uniform}(1, K)$ /* randomly select one site from the candidate set $V$ */
27: submit the job to site $S_s$
28: end procedure
Algorithm 1: The high level idea of the ARA-GLOBAL.

1: procedure ARA-GLOBAL
2:   /* 1. Initialize */
3:   a. the large threshold $thl$ for $K$; /* e.g., $thl = [0.5 \times N]$ */
4:   b. the small threshold $ths$ for $K$; /* e.g., $ths = 1$ */
5:   c. information query delay: $D = d$
6:   /* 2. VM Info Updating */
7:   for each window of $D$ time do
8:     update load $Q_{lens}$ and performance $ServiceRates$ info received from all VMs
9:   end for each
10:   /* 3. Detection of Changes in User Demands */
11:   if Detect the start of burst then
12:     Increase $K$ to $thl$
13:   else if Detect the start of idle then
14:     Decrease $K$ to $ths$
15:   else
16:     Use $K$ for the site selection process
17:   end if
18:   /* 4. Update ExpectedFinishTime for each VM*/
19:   for each VMs do
20:     $ExpectedFinishTime = (NewJobSize + Q_{lens} \times OldServiceRate)/ServiceRate$
21:   end for each
22:   /* 5. Select Among Top-K Shortest ExpectedFinishTime List*/
23:   Sort all VMs $V_i, 1 \leq i \leq N$, by $ExpectedFinishTime$ List
24:   set $V = V_1, V_2, ..., S_K$ /*Get K sites with least load*/
25:   set $s = uniform(1, K)$ /* randomly select one site from the candidate set $V$ */
26:   submit the job to site $S_s$
Content

1. Architecture
2. Model
3. Algorithm
4. Develop
Benchmarking/Sampling Performance to ServiceRate

Arriving Tasks $\lambda$ to Front-End Dispatcher

Amount of Tasks

$\lambda$ uniform $(1,k)$

- VM1: $\mu_1' = 0.5\mu_1$
- VM2: $\mu_2' = 1.33\mu_2$
- VM3: $\mu_3' = 3\mu_3$

Update Queue Length

Expected Qlen = 6 TimeSlots

Expected Qlen = 5 TimeSlots

Expected Qlen = 7 TimeSlots

Top-K

(3) Make Decision On Top-K Candidates Based On Expected Queue Length

Amount of Tasks
Algorithm 1 The high level idea of the ARA-GLOBAL.

1: procedure ARA-GLOBAL
2:  /* 1. Initialize */
3: a. the large threshold $thl$ for $K$; /* e.g., $thl = [0.5 \times N]$ */
4: b. the small threshold $ths$ for $K$; /* e.g., $ths = 1$ */
5: c. information query delay: $D = d$
6: /* 2. VM Info Updating */
7: for each window of $D$ time do
8:    send queries to all computing sites for load information
9:    update load $Qlen$ and performance $ServiceRates$ info received from all VMs
10: end for each
11: /* 3. Detection of Changes in User Demands */
12: if Detect the start of burst then
13:    Increase $K$ to $thl$
14: else if Detect the start of idle then
15:    Decrease $K$ to $ths$
16: else
17:    Use $K$ for the site selection process
18: end if
19: /* 4. Update ExpectedFinishTime for each VM*/
20: for each VMs do
21:    $ExpectedFinishTime = (NewJobSize + Qlen \times OldServiceRate) / ServiceRate$
22: end for each
23: /* 5. Select Among Top-$K$ Shortest ExpectedFinishTime List*/
24: Sort all VMs $V_i, 1 \leq i \leq N$, by $ExpectedFinishTime$ List
25: set $V = V_1, V_2, ..., S_K$ /*Get K sites with least load*/
26: set $s = uniform(1, K)$ /* randomly select one site from the candidate set $V$ */
27: submit the job to site $S_s$
28: end procedure
ServiceRate Tracker

<table>
<thead>
<tr>
<th>ArA</th>
<th>ArA-Global</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Multiple</td>
</tr>
<tr>
<td>ServiceRate</td>
<td>Homogenous</td>
</tr>
</tbody>
</table>
Both **JobSize(ArrivalRate)** and **ServiceRate** are abstract conceptions, so we need to map both of them to real **VM-related criteria**.
Questions:
If we use a performance metrics score as **ServiceRate**, then how to define **JobSize(ArrivalRate)**?