dit
UPM

Evaluating MSRP and MrsP with the multiprocessor Ravenscar profile

Jorge Garrido
Juan Zamorano
Alejandro Alonso
Juan A. de la Puente

Universidad Politécnica de Madrid (UPM), Spain

Have you ever read/written a paper including…

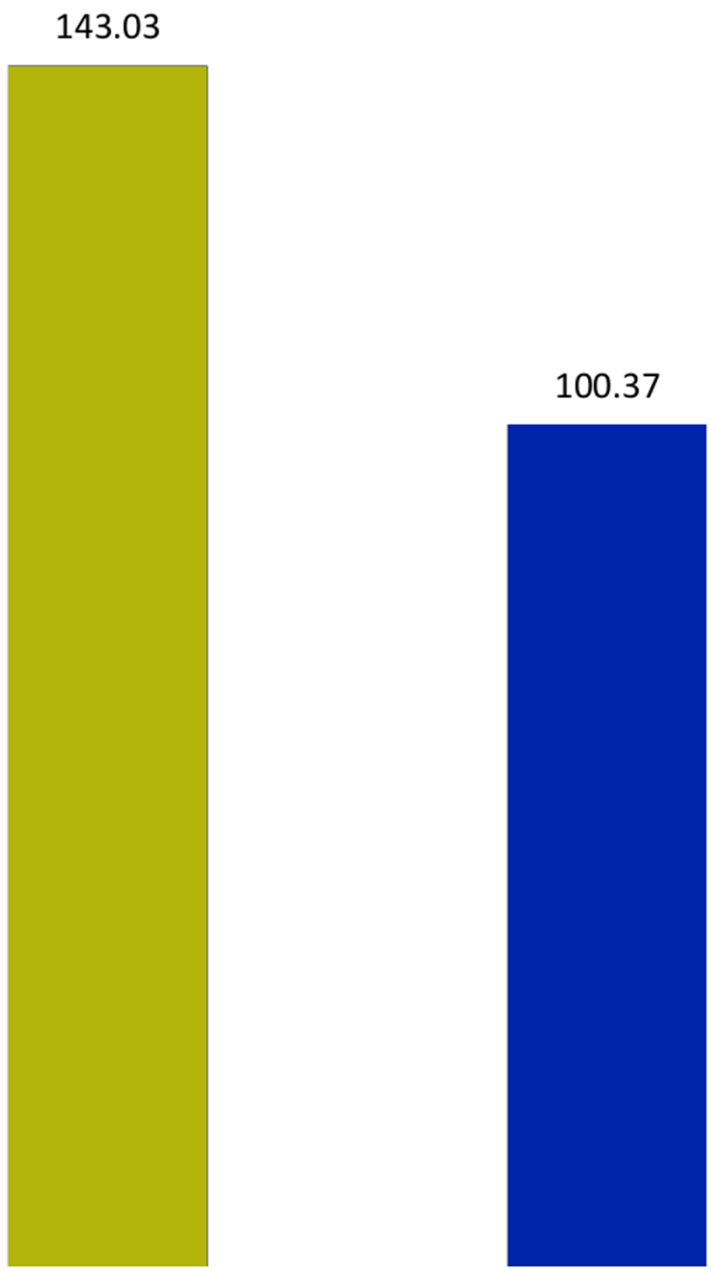
“ for the ease of presentation we omit […]
this would be easily […]”

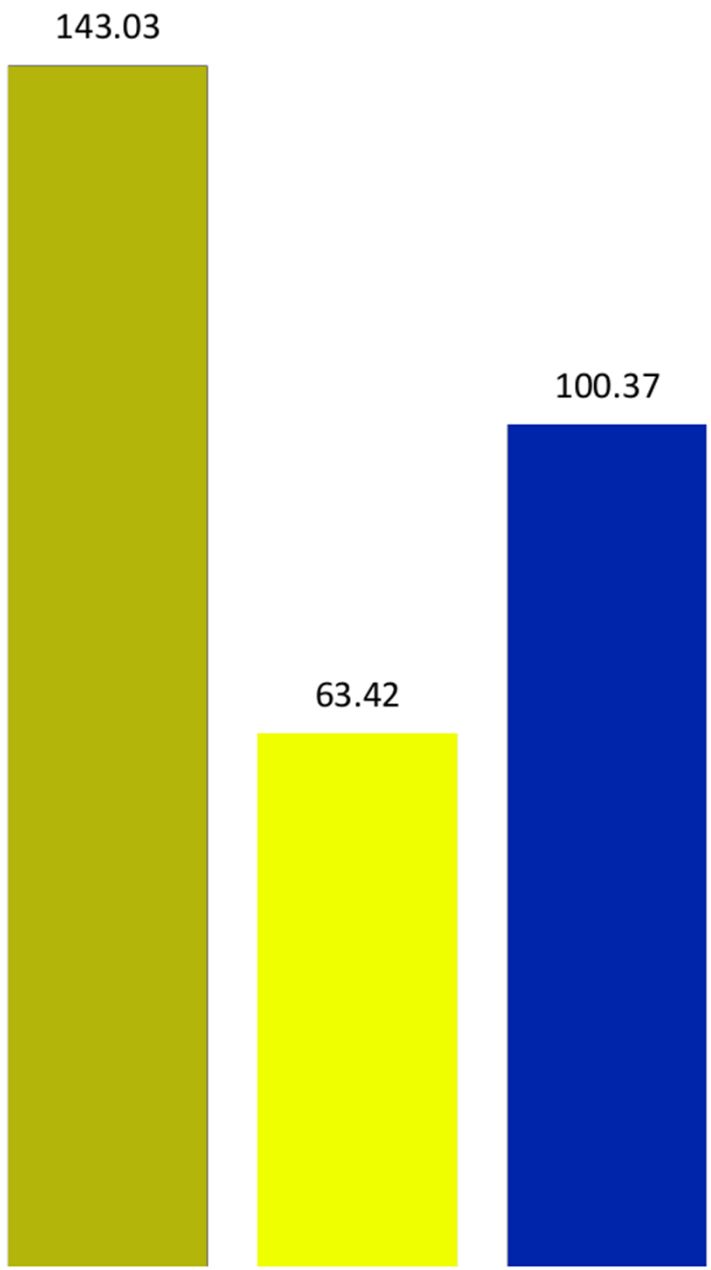
Have you ever read/written a paper including…

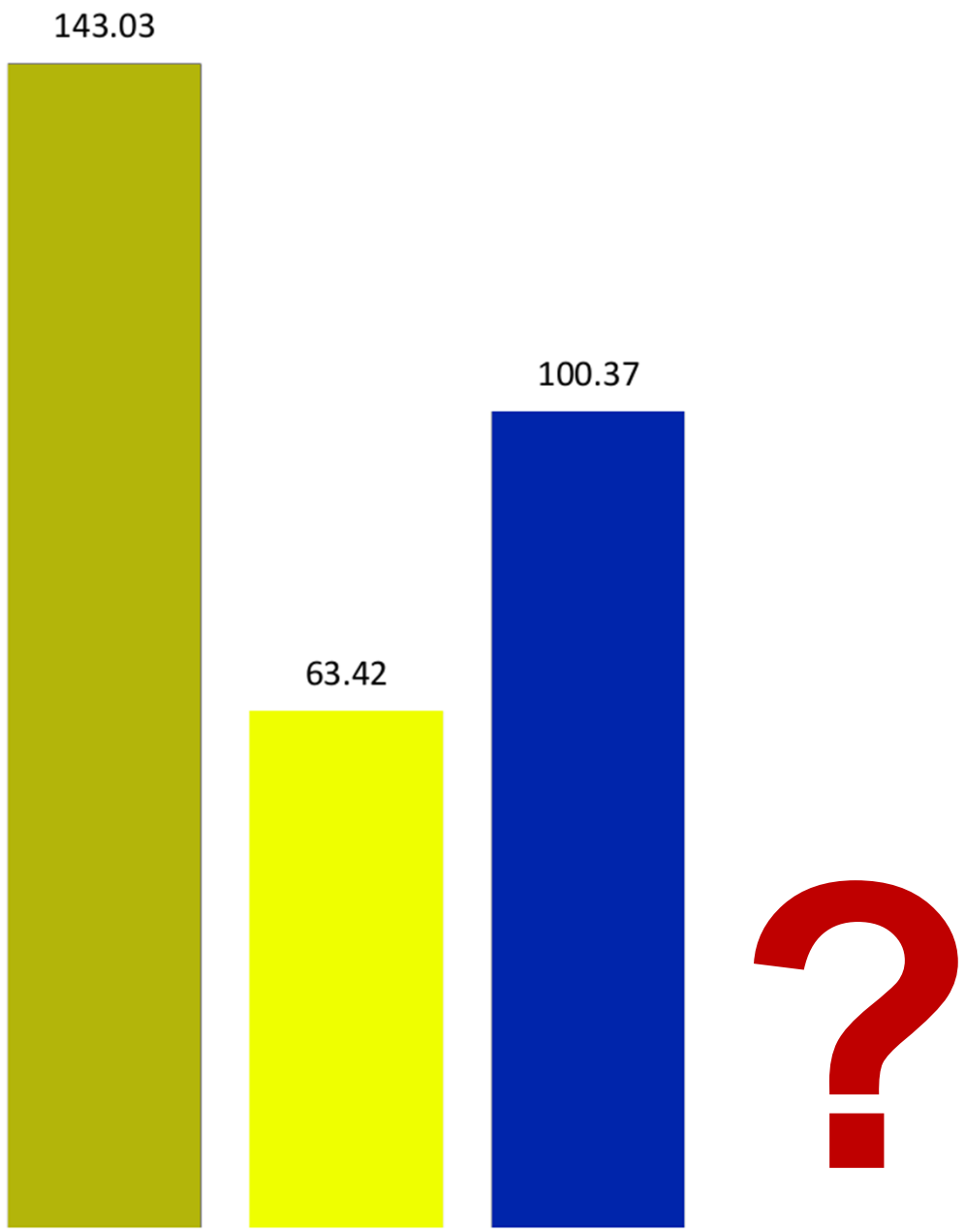
“ for the ease of the presentation we omit […]
this would be easily […]”

… and never read/written about that omitted thing?


143.03


143.03
100.37


143.03
63.42
100.37


143.03
63.42
100.37
?

Have you ever developed a RT system including…

non-volatile memory

communication devices

…

Have you ever developed a RT system including…

non-volatile memory

communication devices

…

… and realized the different access times they have depending on the requested operation?

Motivation

Study the impact of considering heterogeneous access costs for multiprocessor resource sharing protocols

Introduction – MSRP & MrsP

- MSRP – Multiprocessor Stack Resource Prolicy

Introduction – MSRP & MrsP

- MSRP – Multiprocessor Stack Resource Prolicy
- MrsP – Multiprocessor Resource Sharing Protocol

Introduction – MSRP & MrsP

- MSRP – **Multiprocessor** Stack Resource Prolicy
- MrsP – **Multiprocessor** Resource Sharing Protocol

  ✓ Multiprocessor

Introduction – MSRP & MrsP

- MSRP – Multiprocessor Stack Resource Prolicy
- MrsP – Multiprocessor Resource Sharing Protocol

  - ✓ Multiprocessor
  - ✓ Fixed priority scheduling

Introduction – MSRP & MrsP

- MSRP – Multiprocessor Stack **Resource** Prolicy
- MrsP – Multiprocessor **Resource** Sharing Protocol

  - ✓ Multiprocessor
  - ✓ Fixed priority scheduling
  - ✓ Resource arbitration

Introduction – MSRP & MrsP

- MSRP – Multiprocessor Stack **Resource** Prolicy
- MrsP – Multiprocessor **Resource** Sharing Protocol

  - ✓ Multiprocessor
  - ✓ Fixed priority scheduling
  - ✓ Resource arbitration
  - ✓ Access cost to shared resources bounded

Introduction – MSRP & MrsP

- MSRP – Multiprocessor Stack **Resource** Prolicy
- MrsP – Multiprocessor **Resource** Sharing Protocol

  - ✓ Multiprocessor
  - ✓ Fixed priority scheduling
  - ✓ Resource arbitration
  - ✓ Access cost to shared resources bounded
    - ✓ Number of concurrent accesses
    - ✓ FIFO order

Introduction – MSRP & MrsP

- MSRP – Multiprocessor Stack **Resource** Prolicy
- MrsP – Multiprocessor **Resource** Sharing Protocol

  - ✓ Multiprocessor
  - ✓ Fixed priority scheduling
  - ✓ Resource arbitration
  - ✓ Access cost to shared resources bounded
    - ✓ Number of concurrent accesses
    - ✓ FIFO order
  - ✓ Spin-wait

Introduction – MSRP & MrsP

- MSRP – Multiprocessor Stack **Resource** Prolicy
- MrsP – Multiprocessor **Resource** Sharing Protocol

  - ✓ Multiprocessor
  - ✓ Fixed priority scheduling
  - ✓ Resource arbitration
  - ✓ Access cost to shared resources bounded
    - ✓ Number of concurrent accesses
    - ✓ FIFO order
  - ✓ Spin-wait
    - × MSRP : non-preemtable
    - × MrsP : local ceiling priority

Introduction - MrsP

- MrsP – Multiprocessor Resource Sharing Protocol

Introduction - MrsP

- MrsP – Multiprocessor Resource Sharing Protocol
  - Spin- wait is at local ceiling priority of resource

Introduction - MrsP

- MrsP – Multiprocessor Resource Sharing Protocol

  - Spin- wait is at local ceiling priority of resource
    - Preemptable

Introduction - MrsP

- MrsP – Multiprocessor Resource Sharing Protocol

  - Spin- wait is at local ceiling priority of resource
    - Preemptable
    - Local preemptions may affect remote waiting tasks

Introduction - MrsP

- MrsP – Multiprocessor Resource Sharing Protocol

  - Spin- wait is at local ceiling priority of resource
    - Preemptable
    - Local preemptions may affect remote waiting tasks

  - Helping mechanism

Introduction - MrsP

- MrsP – Multiprocessor Resource Sharing Protocol

  - Spin- wait is at local ceiling priority of resource
    - Preemptable
    - Local preemptions may affect remote waiting tasks

  - Helping mechanism
    - Spin-waiting tasks can undertake preempted accesses

Introduction - MrsP

- MrsP – Multiprocessor Resource Sharing Protocol

  - Spin- wait is at local ceiling priority of resource
    - Preemptable
    - Local preemptions may affect remote waiting tasks

  - Helping mechanism
    - Spin-waiting tasks can undertake preempted accesses
    - Progress is done as long as a requesting task can execute

Introduction - MrsP

- MrsP – Multiprocessor Resource Sharing Protocol

  - Spin- wait is at local ceiling priority of resource
    - Preemptable
    - Local preemptions may affect remote waiting tasks

  - Helping mechanism
    - Spin-waiting tasks can undertake preempted accesses
    - Progress is done as long as a requesting task can execute
    - Implemented by migrating the preempted task

Introduction - MrsP

- MrsP – Multiprocessor Resource Sharing Protocol

  - Spin- wait is at local ceiling priority of resource
    - Preemptable
    - Local preemptions may affect remote waiting tasks

  - Helping mechanism
    - Spin-waiting tasks can undertake preempted accesses
    - Progress is done as long as a requesting task can execute
    - Implemented by migrating the preempted task

      × Not allowed under Ravenscar profile

MSRP & MrsP timing analysis

$$R_i = C_i + \max(\hat{e}, \hat{b}) + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

MSRP & MrsP timing analysis

$$R_i = C_i + \max(\hat{e}, \hat{b}) + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$C_i = WCET_i + \sum_{r^j \in \mathbf{F}(\tau_i)} n_i e^j$$

MSRP & MrsP timing analysis

$$R_i = C_i + \max(\hat{e}, \hat{b}) + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$C_i = WCET_i + \sum_{r^j \in \mathbf{F}(\tau_i)} n_i e^j$$

$$e^j = |map(G(r^j))| c^j$$

MSRP & MrsP timing analysis

$$R_i = C_i + \max(\hat{e}, \hat{b}) + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

- Only difference : arrival blocking (ê)

MSRP & MrsP timing analysis

$$R_i = C_i + \max(\hat{e}, \hat{b}) + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

- Only difference : arrival blocking (ê)
  - MSRP: highest e^j value of any resource accessed by a lower priority task.

MSRP & MrsP timing analysis

$$R_i = C_i + \max(\hat{e}, \hat{b}) + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

- Only difference : arrival blocking (ê)
  - MSRP: highest e^j value of any resource accessed by a lower priority task.
  - MrsP : highest e^j value of any resource accessed by a lower priority task and an equal or higher priority task.

MSRP & MrsP timing analysis

$$R_i = C_i + \max(\hat{e}, \hat{b}) + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$C_i = WCET_i + \sum_{r^j \in \mathbf{F}(\tau_i)} n_i e^j$$

$$e^j = |map(G(r^j))| c^j$$

MSRP & MrsP timing analysis

$$R_i = C_i + \max(\hat{e}, \hat{b}) + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$C_i = WCET_i + \sum_{r^j \in \mathbf{F}(\tau_i)} n_i e^j$$

$$e^j = |map(G(r^j))| c^j$$

MSRP & MrsP timing analysis

$$R_i = C_i + \max(\hat{e}, \hat{b}) + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$C_i = WCET_i + \sum_{r^j \in \mathbf{F}(\tau_i)} n_i e^j$$

~~$e^j = |map(G(r^j))| e^j$~~

MSRP & MrsP timing analysis

$$R_i = C_i + \max(\hat{e}, \hat{b}) + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$C_i = WCET_i + \sum_{r^j \in \mathbf{F}(\tau_i)} n_i e^j$$

~~$e^j = |map(G(r^j))| e^j$~~

$$e^j = \sum_{p_k} \hat{c}_k^j$$

MSRP & MrsP timing analysis

$$R_i = C_i + \max(\hat{e}, \hat{b}) + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$C_i = WCET_i + \sum_{r^j \in \mathbf{F}(\tau_i)} n_i e^j$$

~~$e^j = |map(G(r^j))| e^j$~~

$$e^j = \sum_{p_k} \hat{c}_k^j$$

MSRP & MrsP timing analysis

$$R_i = C_i + \max(\hat{e}, \hat{b}) + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$C_i = WCET_i + \sum_{r^j \in \mathbf{F}(\tau_i)} n_i e^j$$

~~$e^j = |map(G(r^j))| e^j$~~

~~$e^j - \sum_{p_k} \hat{c}_k^j$~~

MSRP & MrsP timing analysis

$$R_i = C_i + \max(\hat{e}, \hat{b}) + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$C_i = WCET_i + \sum_{r^j \in \mathbf{F}(\tau_i)} n_i e^j$$

~~$e^j = |map(G(r^j))| e^j$~~

~~$e^j - \sum_{p_k} \hat{c}_k^j$~~

$$e_i^j = c_i^j + \sum_{p_k \setminus \mathbf{P}(\tau_i)} \hat{c}_k^j$$

MSRP & MrsP timing analysis

$$R_i = C_i + \max(\hat{e}, \hat{b}) + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$C_i = WCET_i + \sum_{r^j \in \mathbf{F}(\tau_i)} n_i e^j$$

~~$e^j = |map(G(r^j))| e^j$~~

~~$e^j - \sum_{p_k} \hat{c}_k^j$~~

$$e_i^j = c_i^j + \sum_{p_k \setminus \mathbf{P}(\tau_i)} \hat{c}_k^j$$

MSRP & MrsP timing analysis

$$R_i = C_i + \max(\hat{e}, \hat{b}) + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$C_i = WCET_i + \sum_{r^j \in \mathbf{F}(\tau_i)} n_i e^j$$

~~$e^j = |map(G(r^j))| e^j$~~

~~$e^j - \sum_{p_k} \hat{c}_k^j$~~

~~$e_i^j = c_i^j + \sum_{p_k \setminus \mathbf{P}(\tau_i)} \hat{c}_k^j$~~

MSRP & MrsP timing analysis

$$R_i = C_i + \max(\hat{e}, \hat{b}) + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$C_i = WCET_i + \sum_{r^j \in \mathbf{F}(\tau_i)} n_i e^j$$

~~$e^j = |map(G(r^j))| e^j$~~

~~$e^j - \sum_{p_k} \hat{c}_k^j$~~ $\quad e_{i,n}^j = c_{i,n}^j + \sum_{p_k \setminus \mathbf{P}(\tau_i)} \hat{c}_k^j$

~~$e_i^j = c_i^j + \sum_{p_k \setminus \mathbf{P}(\tau_i)} \hat{c}_k^j$~~

MSRP & MrsP timing analysis

$$R_i = C_i + \max(\hat{e}, \hat{b}) + \sum_{\tau_j \in \mathbf{hpl}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$C_i = WCET_i + \sum_{r^j \in \mathbf{F}(\tau_i)} n_i e^j$$

$$e^j_{i,n} = c^j_{i,n} + \sum_{p_k \setminus \mathbf{P}(\tau_i)} \hat{c}^j_k$$

Evaluation

- Synthetic task sets

Evaluation

- Synthetic task sets
- Nº of processors (N)

Evaluation

- Synthetic task sets
- Nº of processors (N) = Nº shared resources (R)

Evaluation

- Synthetic task sets
- Nº of processors (N) = Nº shared resources (R): 2, 4, 8, 16

Evaluation

- Synthetic task sets
- Nº of processors (N) = Nº shared resources (R): 2, 4, 8, 16
- Processor utilization: [0.1, 0.05, 0.7]

Evaluation

- Synthetic task sets
- Nº of processors (N) = Nº shared resources (R): 2, 4, 8, 16
- Processor utilization: [0.1, 0.05, 0.7]
- Task % of processor utilization: random uniformly distributed
  [0.1,0.2], [0.1,0.3], [0.1,0.4], [0.1,0.5], [0.1,0.9], [0.25,0.75]

Evaluation

- Synthetic task sets
- Nº of processors (N) = Nº shared resources (R): 2, 4, 8, 16
- Processor utilization: [0.1, 0.05, 0.7]
- Task % of processor utilization: random uniformly distributed
  [0.1,0.2], [0.1,0.3], [0.1,0.4], [0.1,0.5], [0.1,0.9], [0.25,0.75]
- Priority assignment : Deadline Monotonic

Evaluation

- Synthetic task sets
- Nº of processors (N) = Nº shared resources (R): 2, 4, 8, 16
- Processor utilization: [0.1, 0.05, 0.7]
- Task % of processor utilization: random uniformly distributed
  [0.1,0.2], [0.1,0.3], [0.1,0.4], [0.1,0.5], [0.1,0.9], [0.25,0.75]
- Priority assignment : Deadline Monotonic
- 100,000 task sets for each combination

Evaluation

- Synthetic task sets
- Nº of processors (N) = Nº shared resources (R): 2, 4, 8, 16
- Processor utilization: [0.1, 0.05, 0.7]
- Task % of processor utilization: random uniformly distributed
  [0.1,0.2], [0.1,0.3], [0.1,0.4], [0.1,0.5], [0.1,0.9], [0.25,0.75]
- Priority assignment : Deadline Monotonic
- 100,000 task sets for each combination
  - Each analyzed with MSRP and MrsP
    - homogeneous
    - heterogeneous

Evaluation – Resource usage

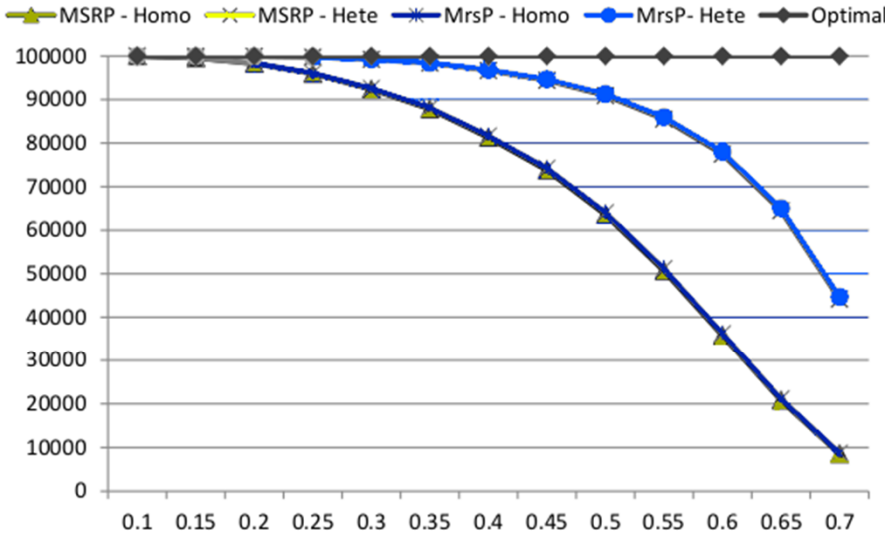
- Each task has probability of 1/R of using each resource

Evaluation – Resource usage

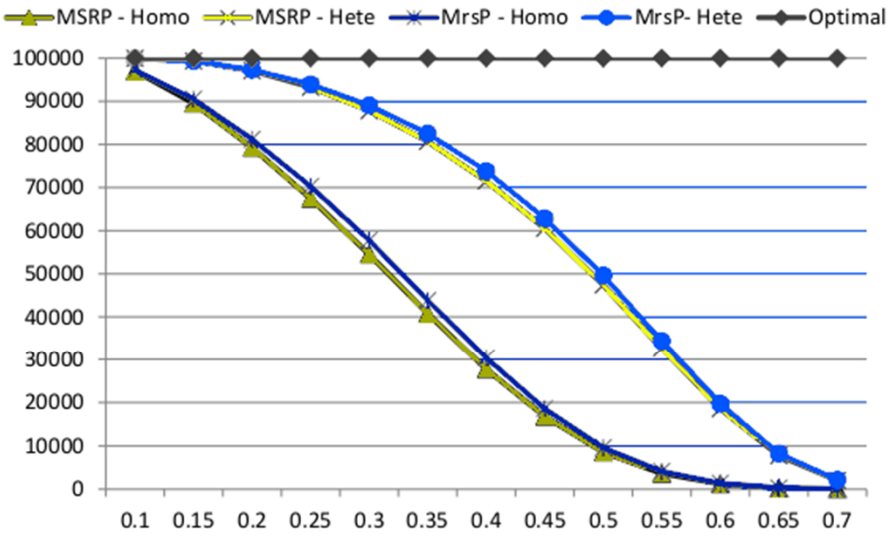
- Each task has probability of 1/R of using each resource
- Accessing time is random [0.01%, 10.00%] of task C

Evaluation – Resource usage

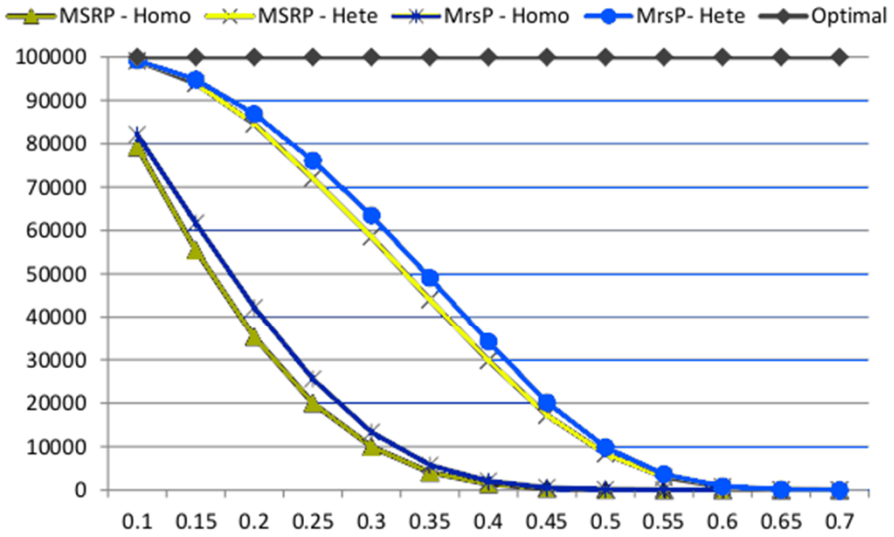
- Each task has probability of 1/R of using each resource
- Accessing time is random [0.01%, 10.00%] of task C
- If the resource is used, the probability of accessing the resource *n* time per release is $P(n) = 1/n^2$



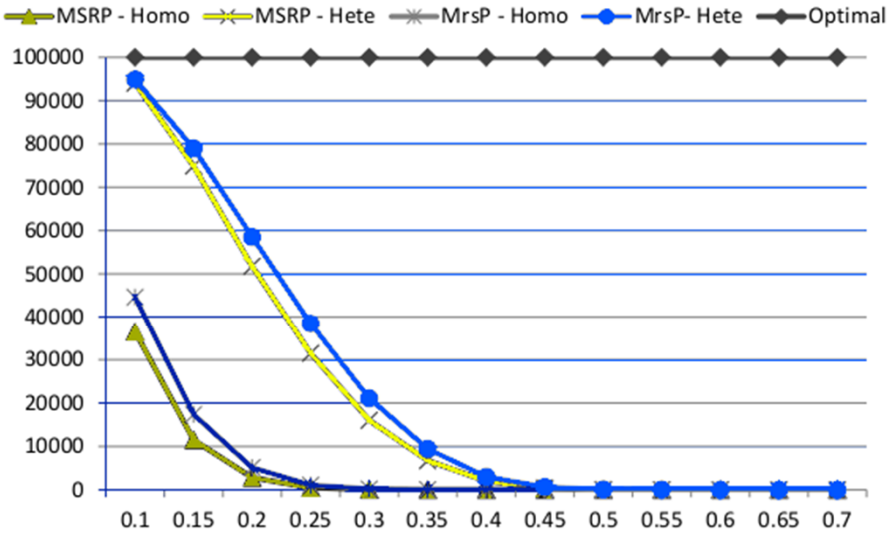
(a) $N = 2$.



(b) $N = 4$.



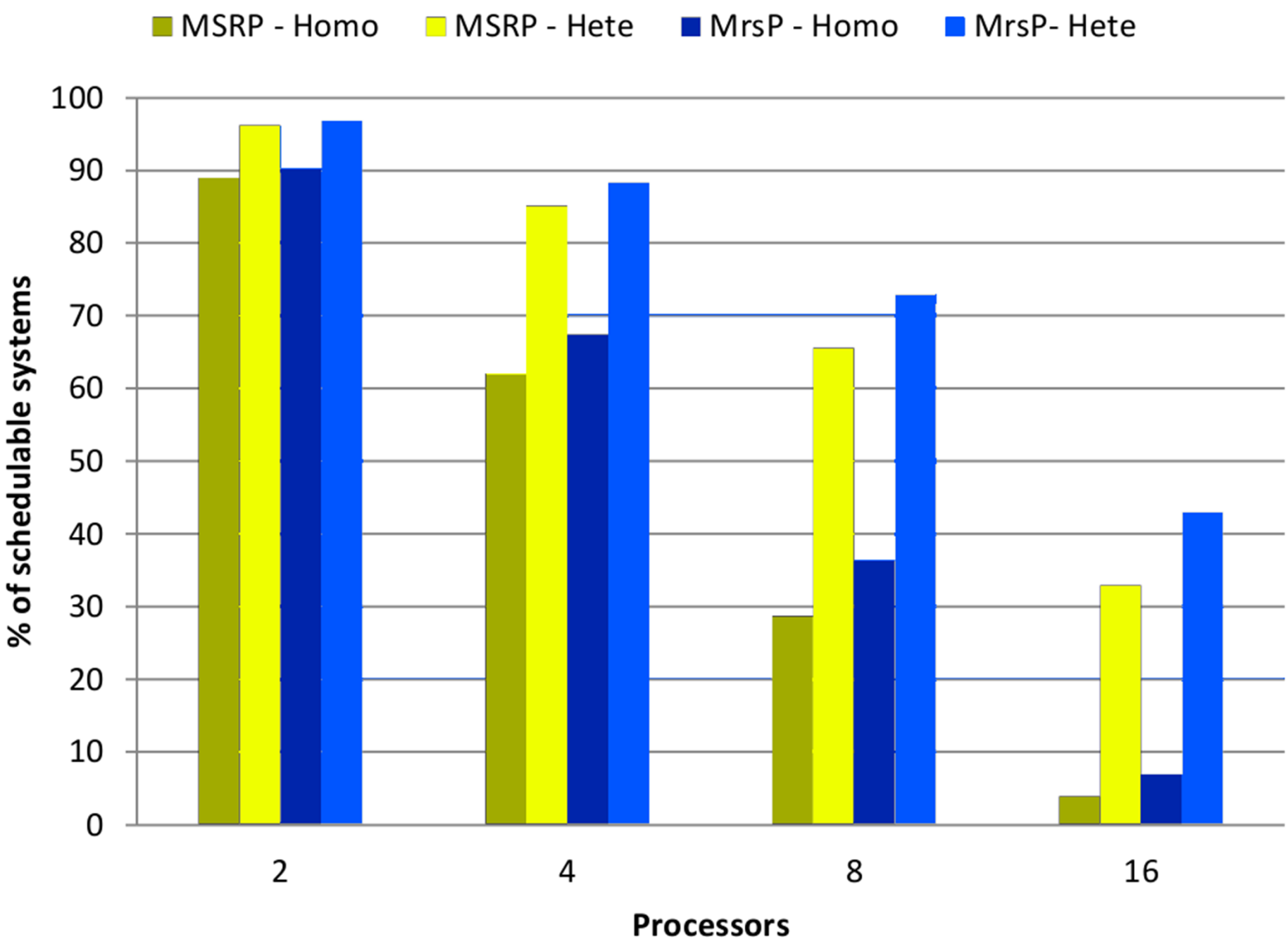
(c) $N = 8$.



(d) $N = 16$.

Schedulable systems per system utilization

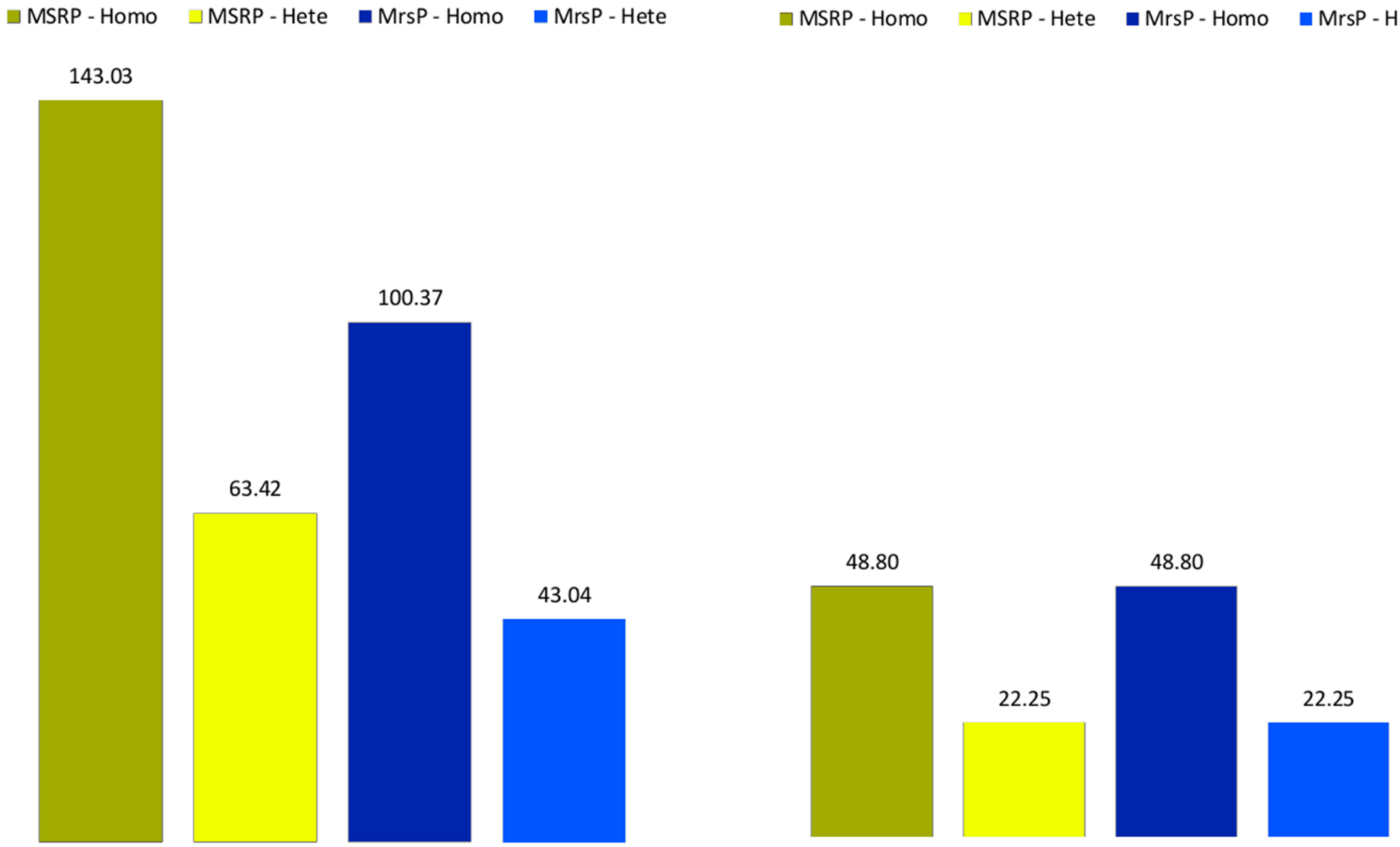
Tasks utilization [0.1,0.2]



Schedulable systems per number of processors

Processor utilization 0.35 Tasks utilization [0.1,0.9]

Response time overhead (%) over optimal resource scheduling



Highest priority tasks

Lowest priority tasks

Processor utilization 0.1 Tasks utilization [0.1,0.9] 8 processors

Evaluation - Tendencies

Evaluation - Tendencies

- High priority tasks more benefited in relative overheads

Evaluation - Tendencies

- High priority tasks more benefited in relative overheads
- Lower priority tasks more benefited in net time values

Evaluation - Tendencies

- High priority tasks more benefited in relative overheads
- Lower priority tasks more benefited in net time values

- MSRP more benefited from heterogeneous approach

Evaluation - Tendencies

- High priority tasks more benefited in relative overheads
- Lower priority tasks more benefited in net time values

- MSRP more benefited from heterogeneous approach
  - Ravenscar-like approach

Evaluation - Tendencies

- High priority tasks more benefited in relative overheads
- Lower priority tasks more benefited in net time values

- MSRP more benefited from heterogeneous approach
  - Ravenscar-like approach

- The more complex the system, the more benefit from heterogeneity

Evaluation - Tendencies

- High priority tasks more benefited in relative overheads
- Lower priority tasks more benefited in net time values

- MSRP more benefited from heterogeneous approach
  - Ravenscar-like approach

- The more complex the system, the more benefit from heterogeneity
  - Number of processors

Evaluation - Tendencies

- High priority tasks more benefited in relative overheads
- Lower priority tasks more benefited in net time values

- MSRP more benefited from heterogeneous approach
  - Ravenscar-like approach

- The more complex the system, the more benefit from heterogeneity
  - Number of processors
  - Processor utilization

Evaluation - Tendencies

- High priority tasks more benefited in relative overheads
- Lower priority tasks more benefited in net time values

- MSRP more benefited from heterogeneous approach
  - Ravenscar-like approach

- The more complex the system, the more benefit from heterogeneity
  - Number of processors
  - Processor utilization
  - Disparity among task utilizations

Evaluation - Tendencies

- High priority tasks more benefited in relative overheads
- Lower priority tasks more benefited in net time values

- MSRP more benefited from heterogeneous approach
  - Ravenscar-like approach

- The more complex the system, the more benefit from heterogeneity
  - Number of processors
  - Processor utilization
  - Disparity among task utilizations

- Raw data, more results & charts available upon request

Conclusions

Conclusions

- “Simple” improvements matter

Conclusions

- "Simple" improvements matter
  - Formalized

Conclusions

- "Simple" improvements matter
  - Formalized
  - Verified

Conclusions

- "Simple" improvements matter
  - Formalized
  - Verified
  - Evaluated

Conclusions

- "Simple" improvements matter
  - Formalized
  - Verified
  - Evaluated

- MrsP strictly dominates MSRP

Conclusions

- "Simple" improvements matter
  - Formalized
  - Verified
  - Evaluated

- MrsP strictly dominates MSRP
  - MSRP : Ravenscar choice

Conclusions

- "Simple" improvements matter
  - Formalized
  - Verified
  - Evaluated

- MrsP strictly dominates MSRP
  - MSRP : Ravenscar choice
  - MrsP : full Ada programs

Conclusions

- "Simple" improvements matter
  - Formalized
  - Verified
  - Evaluated

- MrsP strictly dominates MSRP
  - MSRP : Ravenscar choice
  - MrsP : full Ada programs

- Task allocation strategies required

Evaluating MSRP and MrsP with the multiprocessor Ravenscar profile

Jorge Garrido
Juan Zamorano
Alejandro Alonso
Juan A. de la Puente

str@dit.upm.es

Universidad Politécnica de Madrid (UPM), Spain