Mahmoud Mahmoud
PhD Student
Institute for Radio Astronomy & Space Research (IRASR)
AUT University.

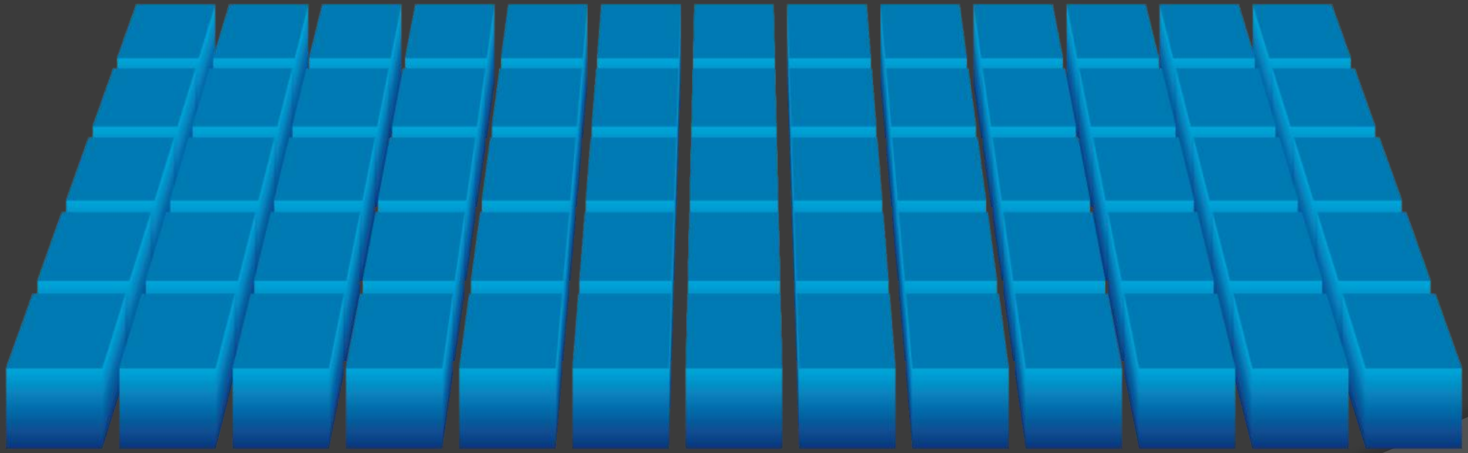
SCHEDULING OPTIMIZATION FOR SKA HPC MIDDLEWARE

Computing for SKA (C4SKA) Colloquium 2015

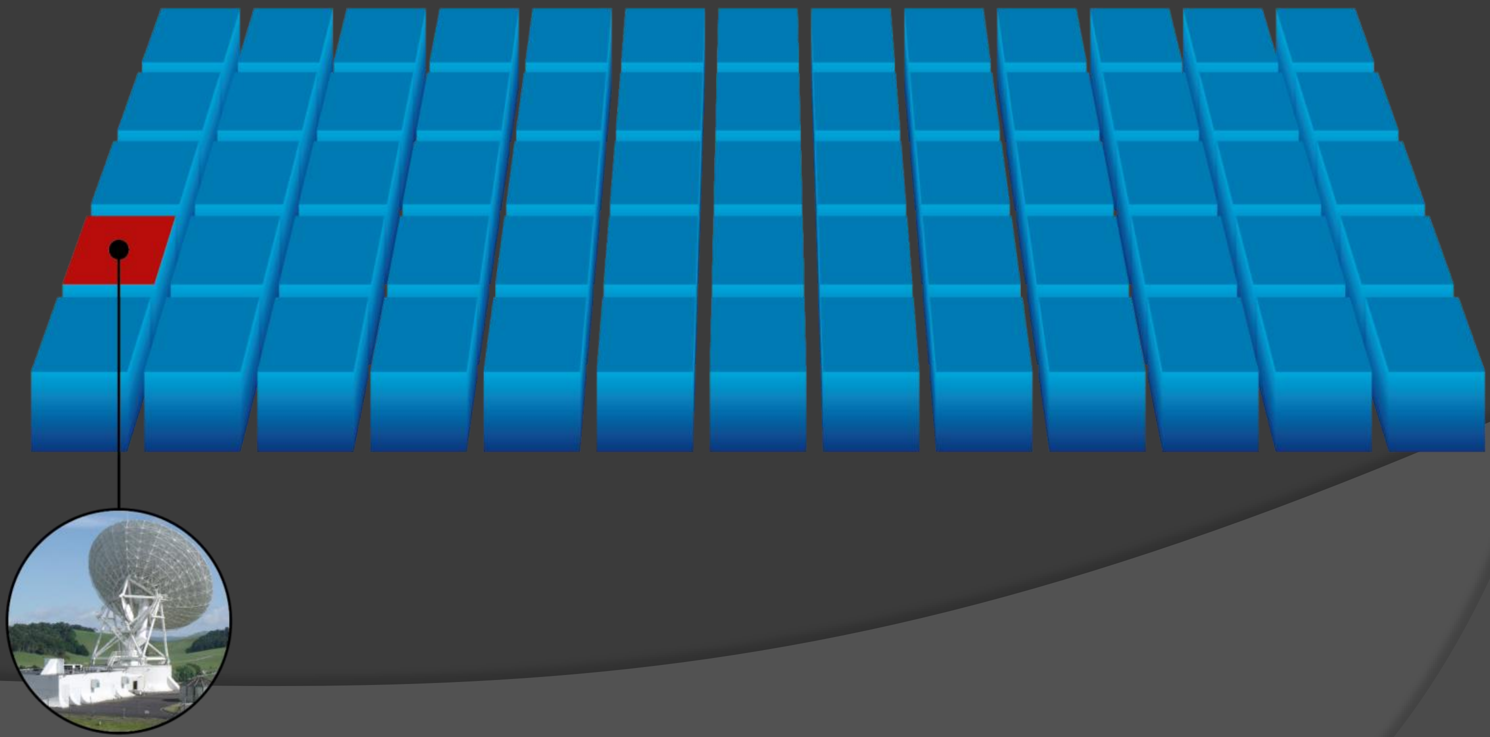
Middleware

- Is software that facilitates combination of autonomous operating environments into a unified operating environment.
- Communication management:
  - Hides network protocols.
  - Common interface for communication.
  - Data type marshaling.
- Resource management:
  - Resource monitoring.
  - Task scheduling.
  - Load balancing.
- Distributed application development:
  - Programming language.
  - Integrated development environment (IDE).
  - Debugging and profiling tools.

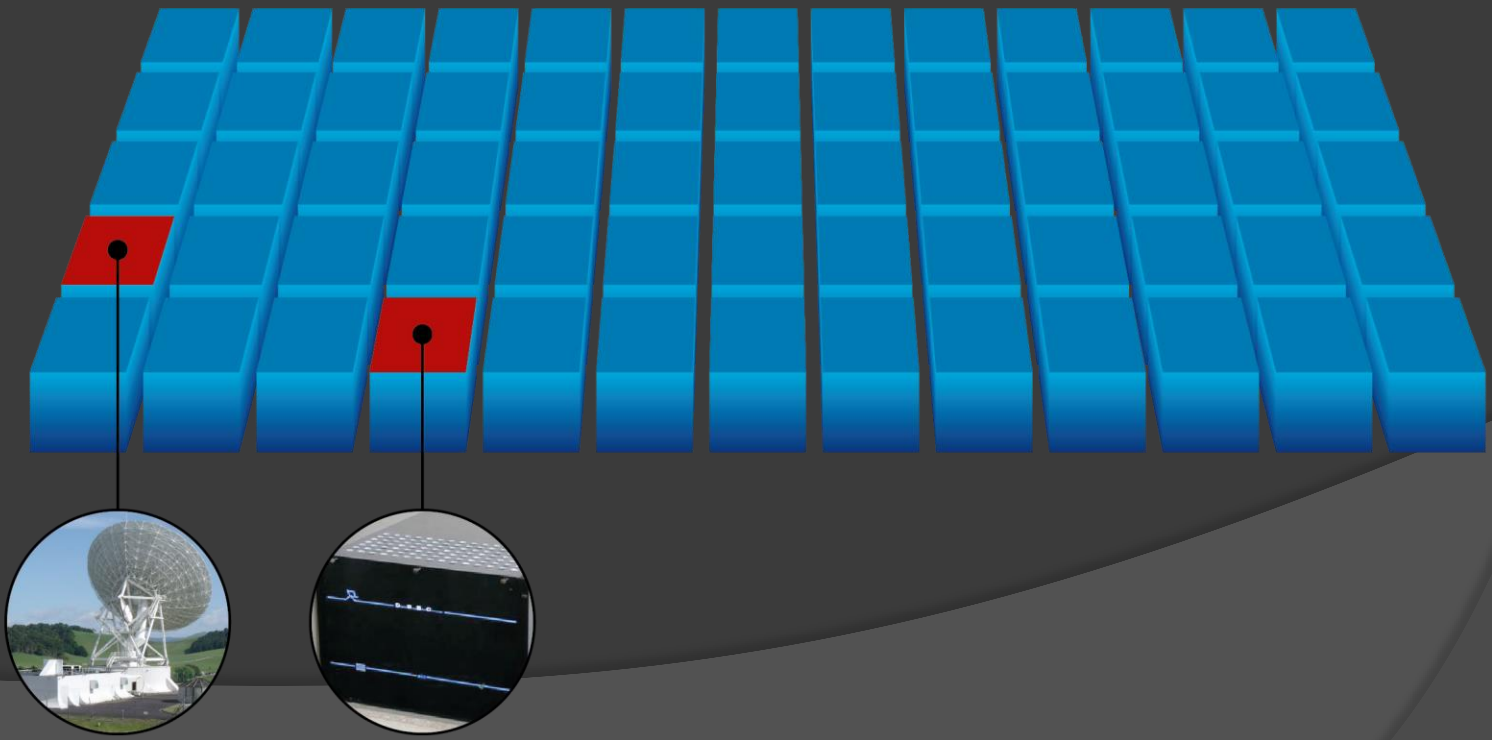
Top-level View for Middleware Placement



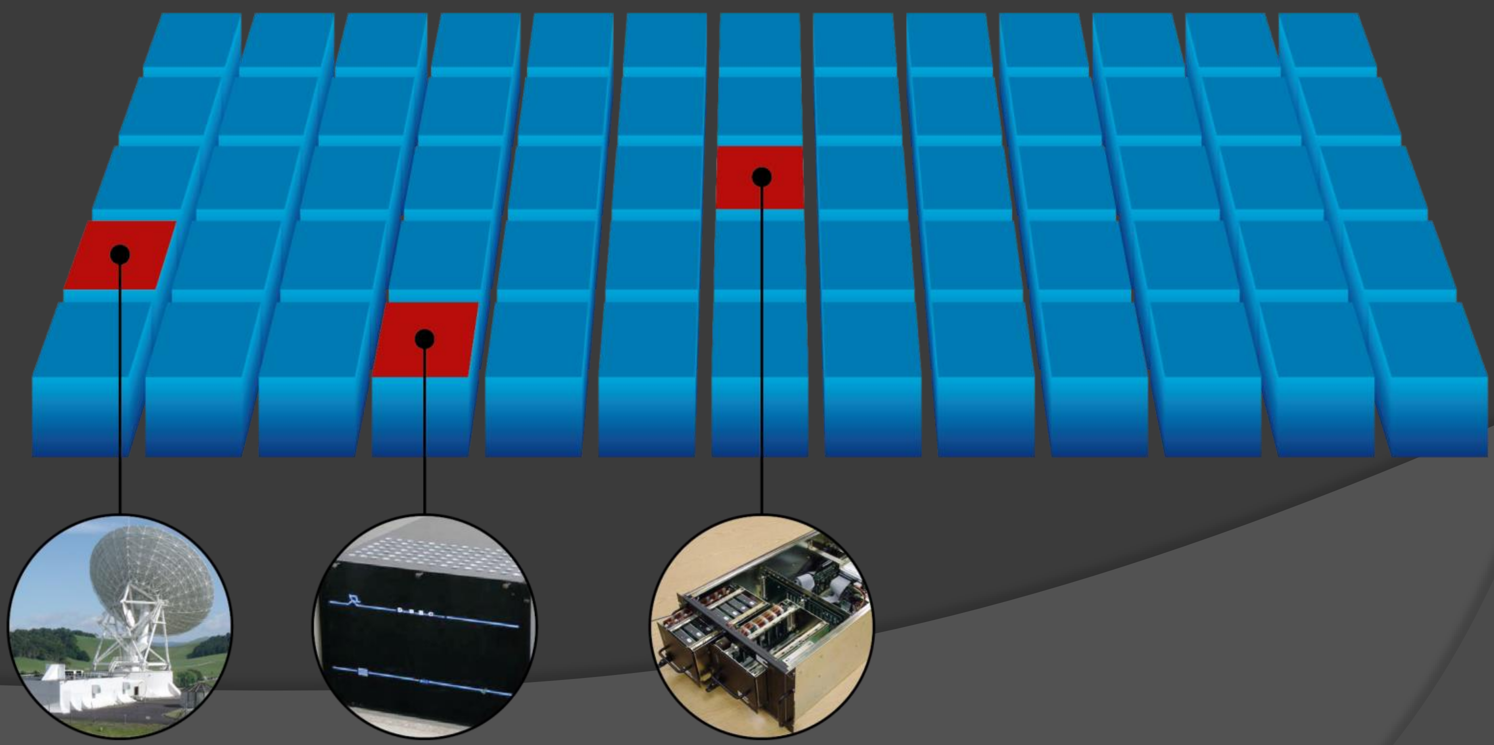
Top-level View for Middleware Placement



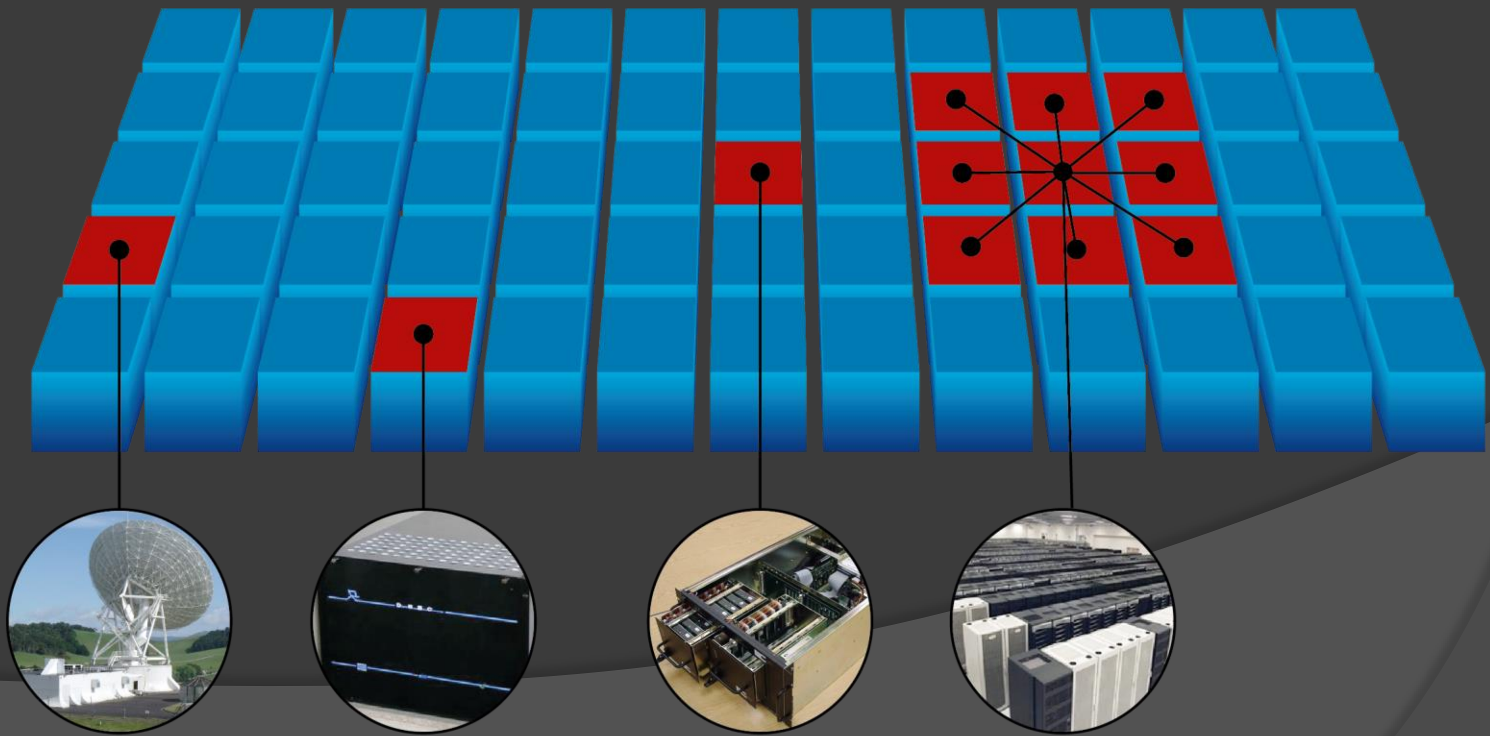
Top-level View for Middleware Placement



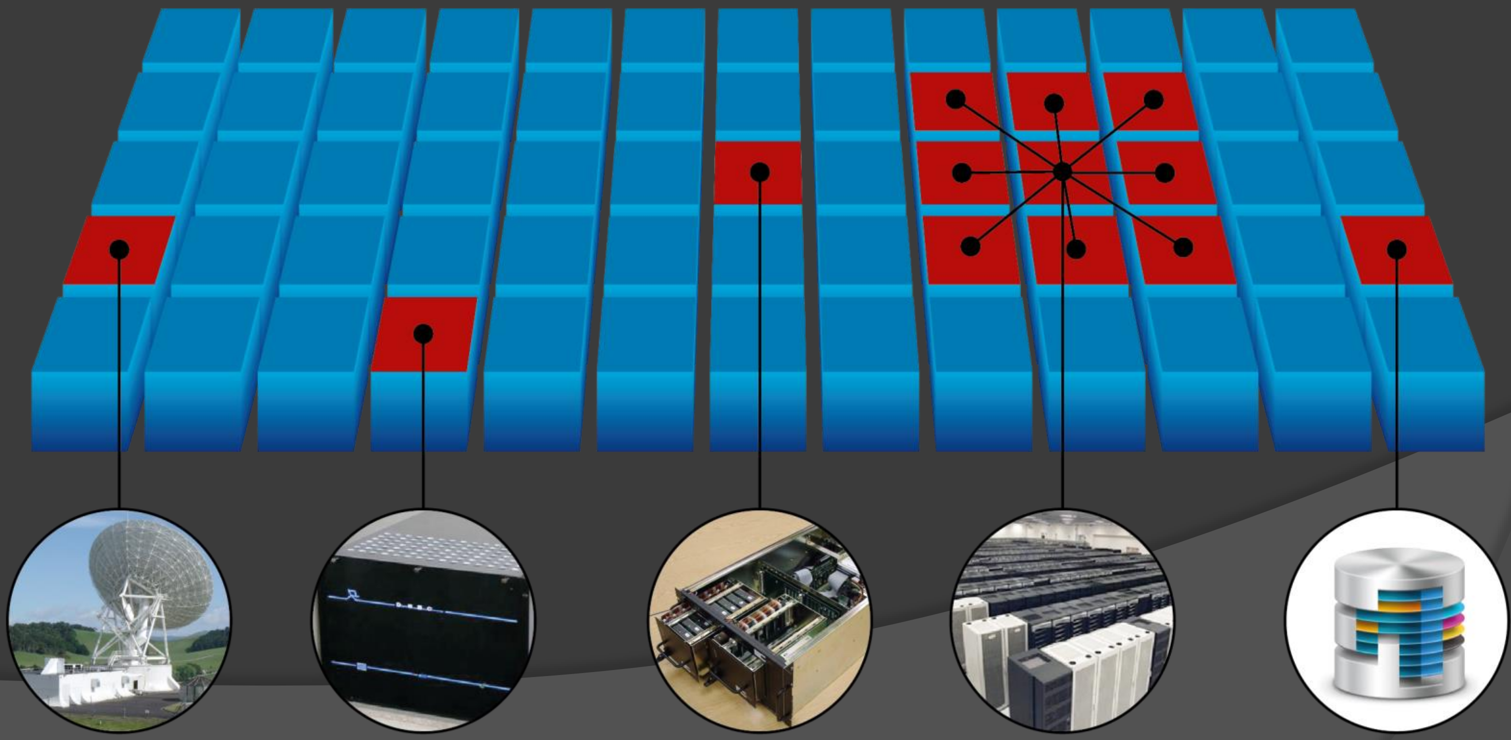
Top-level View for Middleware Placement



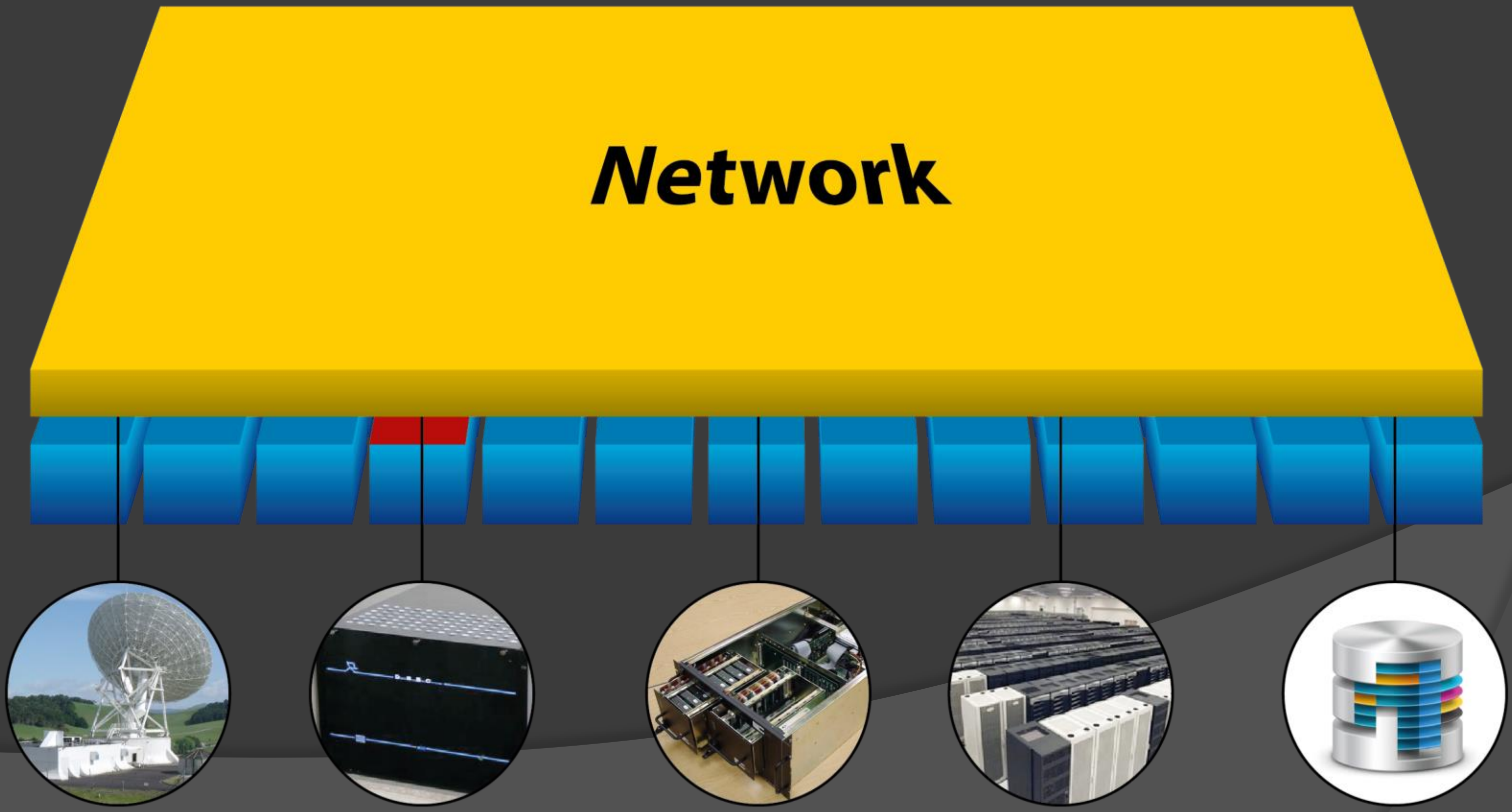
Top-level View for Middleware Placement



Top-level View for Middleware Placement



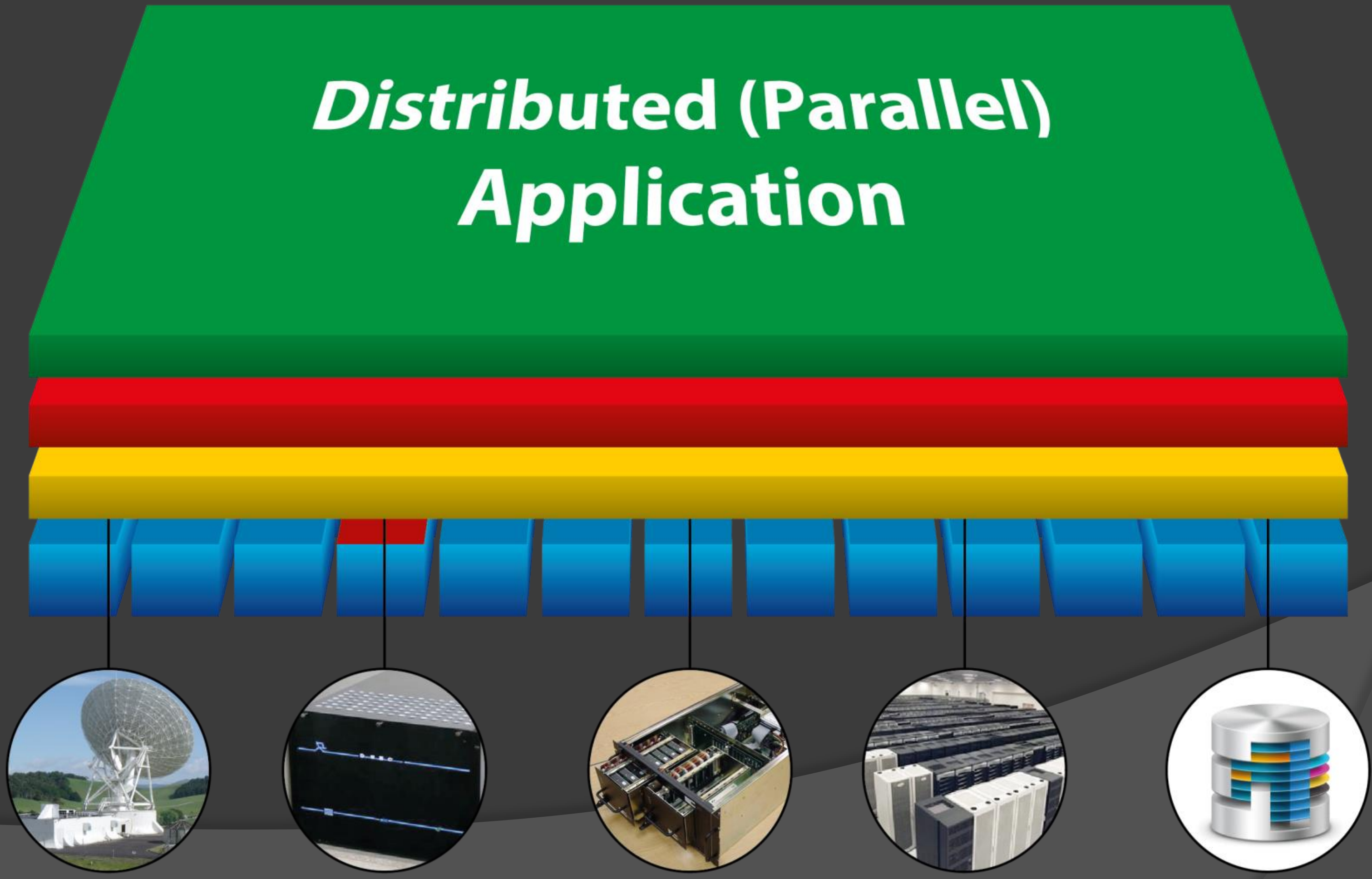
Top-level View for Middleware Placement



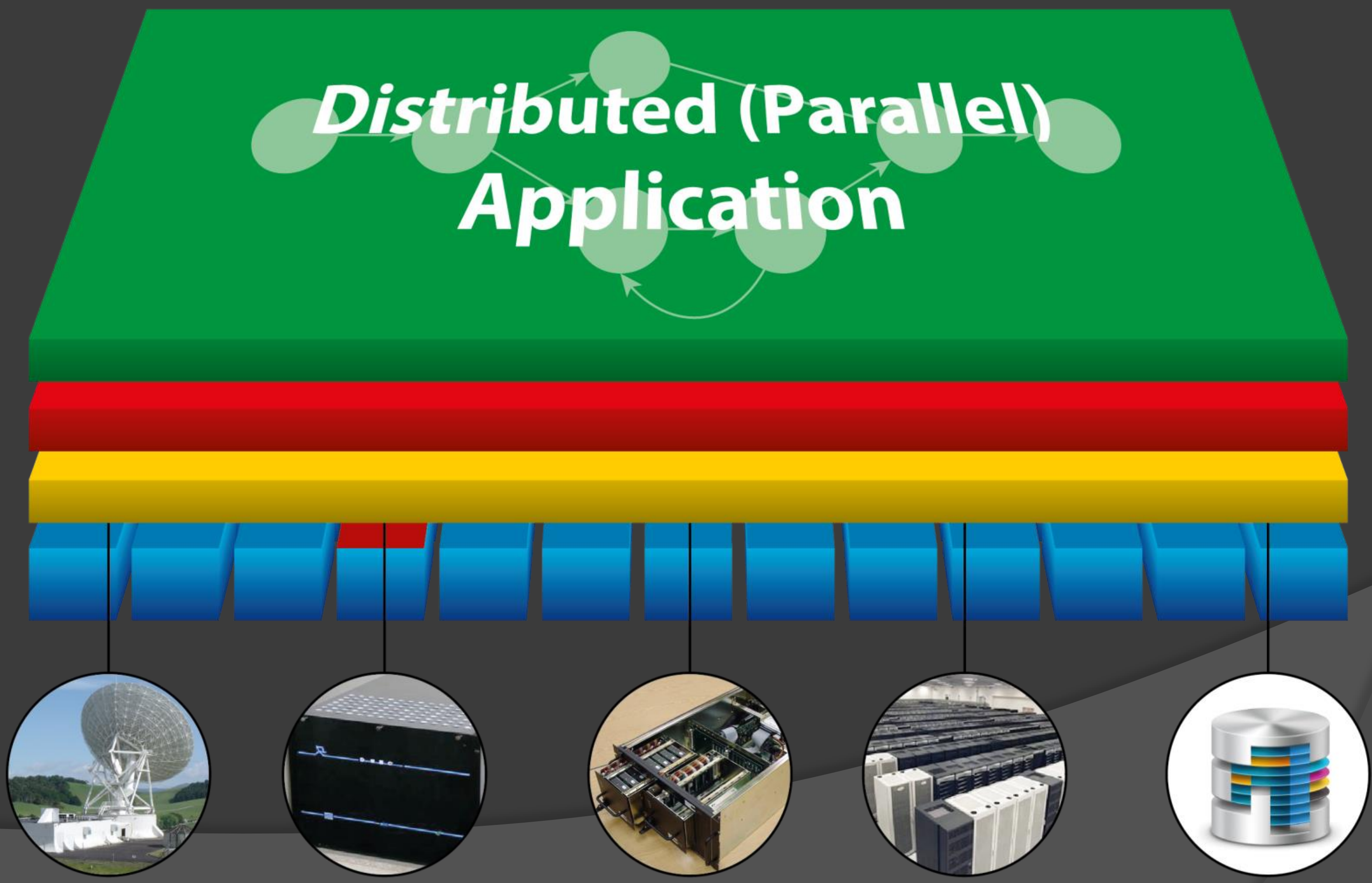
Top-level View for Middleware Placement

Top-level View for Middleware Placement

Top-level View for Middleware Placement



Top-level View for Middleware Placement



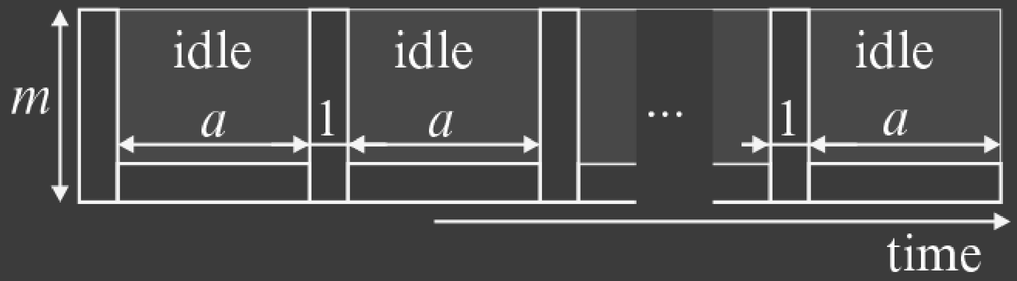
Examples of some HPC Middleware Usage in Radio Astronomy

- DiFX – MPI based parallel software correlator.
- IBM InfoSphere Streams – correlation, RFI mitigation, and imaging.
- ARTEMIS Pelican/Panda – pulsar search pipelines.

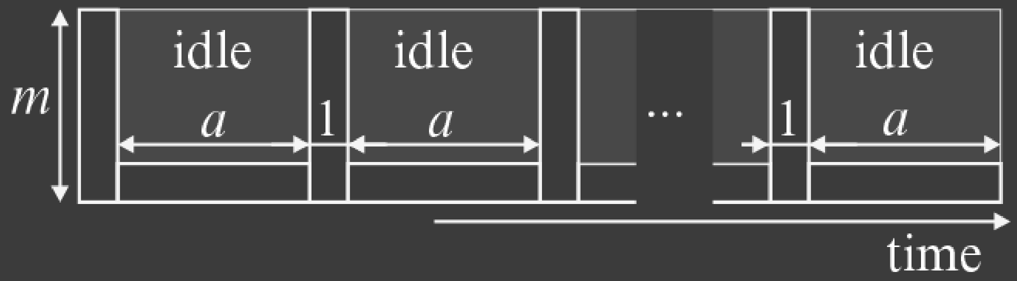
Basic Parallel Processing Scheduling Theory

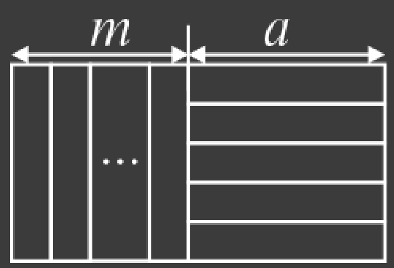
- A parallel application is defined by a set of tasks and communications.
- Tasks may be partially ordered and represented as by a task graph.
- The task graph is referred to as a job.

Importance of Scheduling – an example

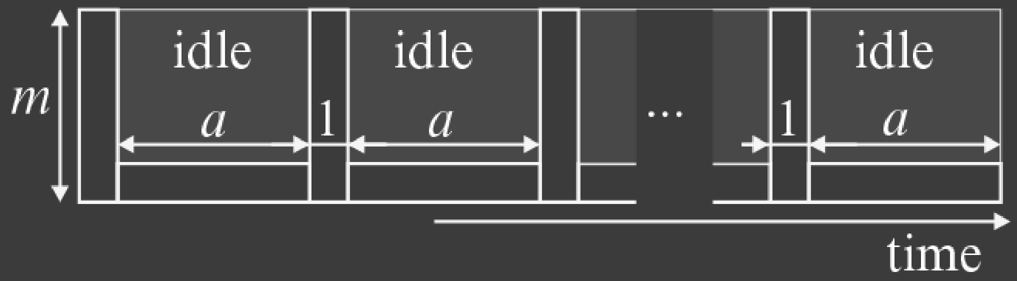


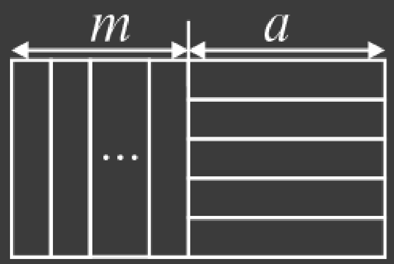
Importance of Scheduling – an example



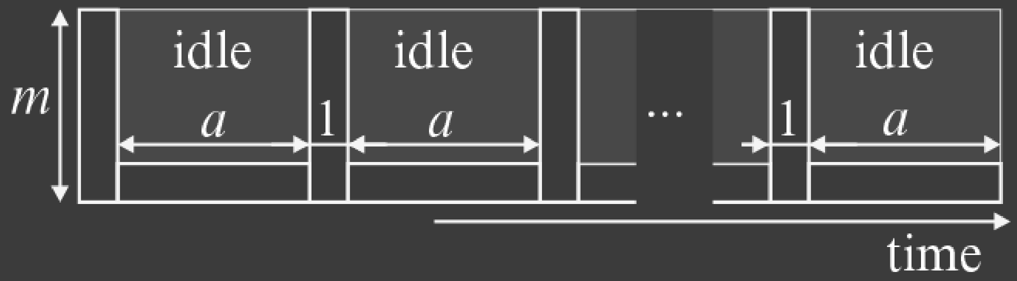


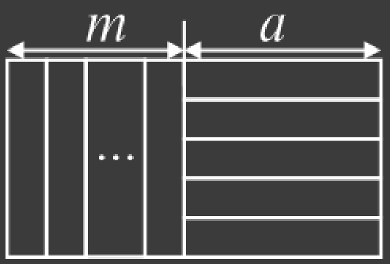
Importance of Scheduling – an example





Importance of Scheduling – an example





$$\frac{m(a+1)}{m+a}$$ Shorter

Classical Scheduling Theory Optimality Criteria

- Schedule length
- Maximum lateness
- Mean tardiness
- Number of late tasks
- Weighted number of late tasks
- Mean flow time
- Mean weighted

SKA “Big Data” Problem Implications on Scheduling

- Big data volumes impractical to store for later processing.
- To avoid mass storage data must remain in motion.
- Data in motion is data streaming and implies pipeline (stream) based processing architecture.
- Power efficiency criteria

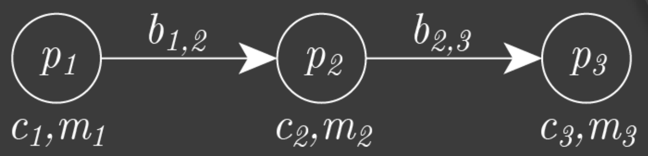
Stream Processing Paradigm

Stream Processing Paradigm

Minimum Power Dissipation Based Scheduling

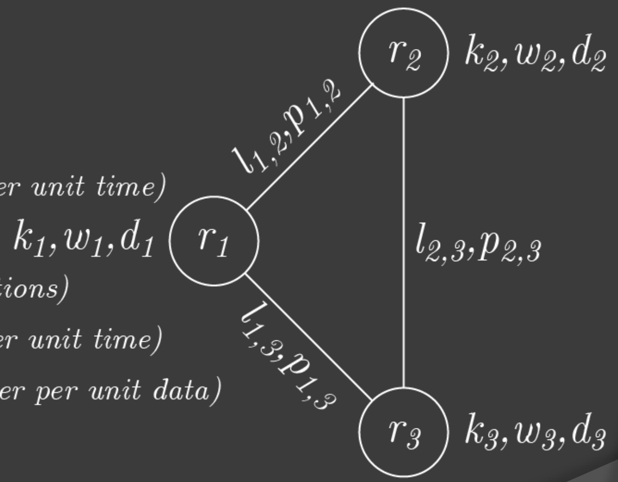
- Dynamic Programming
- Minimal Power Dissipation Path
- Make simplifications to reduce number of combinations
  - Initial conditions
  - Best effort no gaurantee
- Multiple job scheduling
- Adapting to change

Metrics



chained parallel process application:

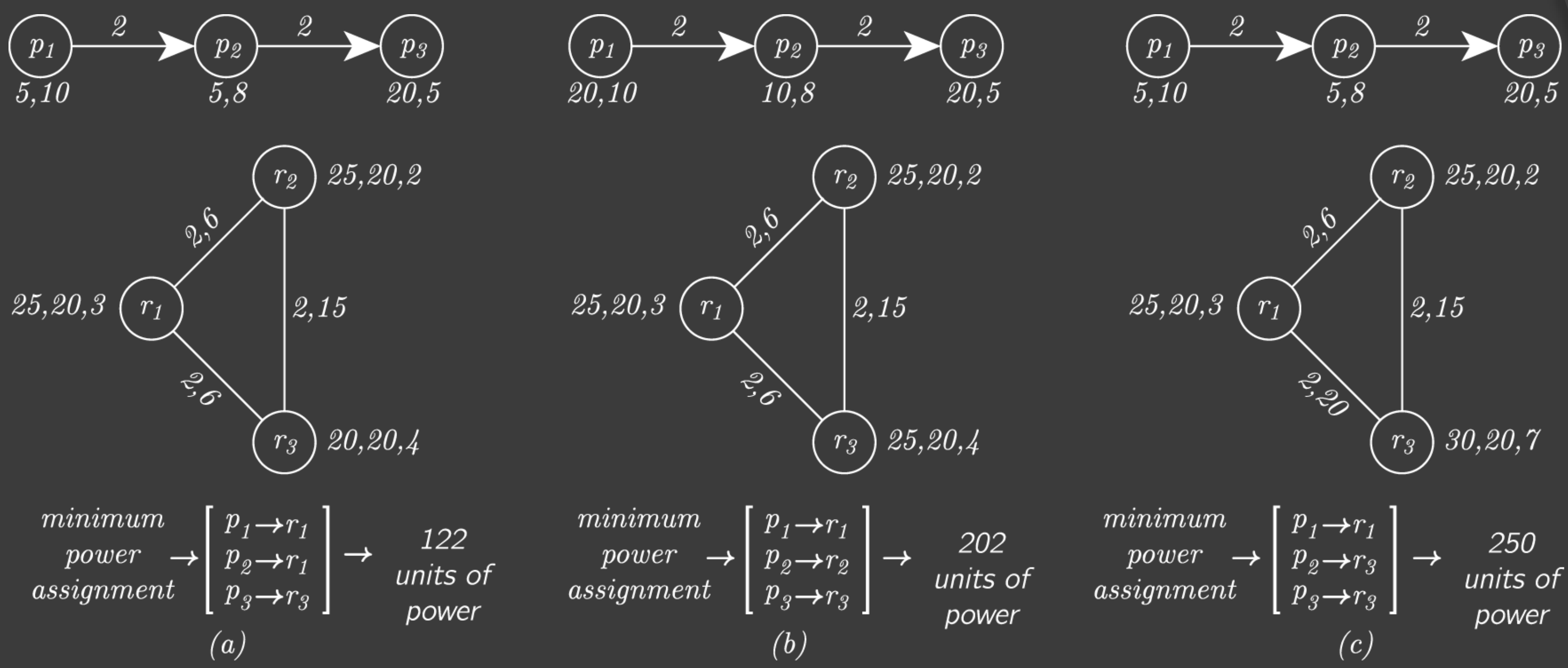
p_i : *process* i *(uuid)*
c_i : *compute requirements for process* i *(unit data operations per unit time)*
m_i : *memory requirements of process* i *(unit data)*
$b_{i,j}$: *output bandwidth from process* i *to process* j *(unit data per unit time)*



distributed computing resource:

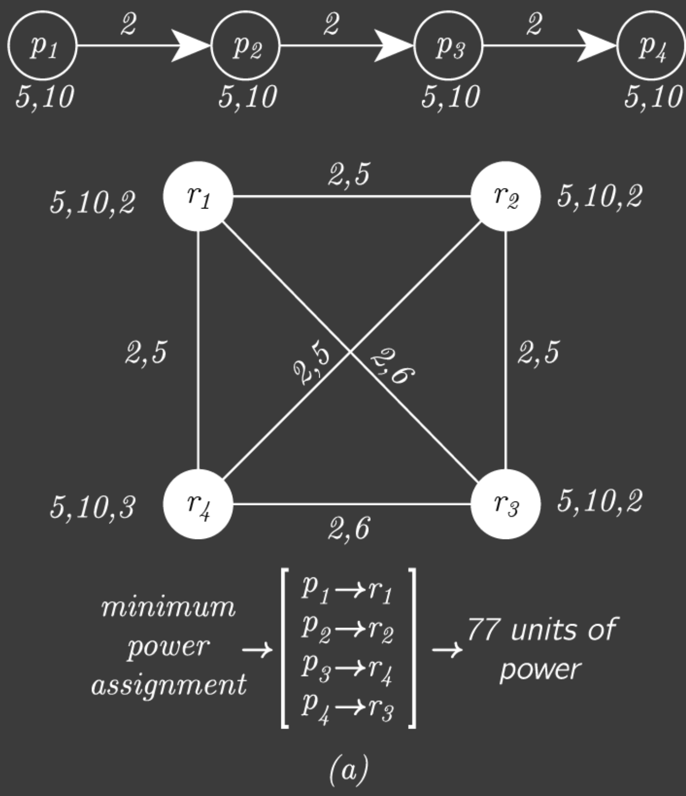
r_s : *processor* s *(uuid)*
k_s : *compute capability of processor* s *(unit data operations per unit time)*
d_s : *memory capacity of processor* s *(unit data)*
w_s : *power usage of processor* s *(unit power per unit data operations)*
$l_{s,t}$: *link bandwidth between processor* s *and* t *(unit data per unit time)*
$p_{s,t}$: *link power usage between processor* s *and* t *(unit power per unit data)*

Example

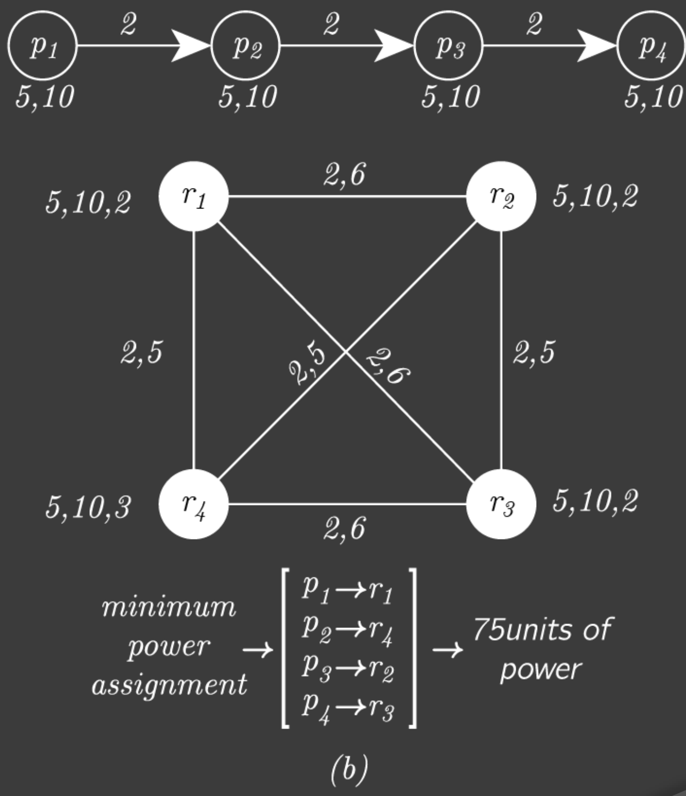


p1 2 p2 2 p3
5,10 5,8 20,5
r2 25,20,2
2,6
25,20,3 r1 2,15
2,6
r3 20,20,4
minimum power assignment → [p1→r1 p2→r1 p3→r3] → 122 units of power
(a)
p1 2 p2 2 p3
20,10 10,8 20,5
r2 25,20,2
2,6
25,20,3 r1 2,15
2,6
r3 25,20,4
minimum power assignment → [p1→r1 p2→r2 p3→r3] → 202 units of power
(b)
p1 2 p2 2 p3
5,10 5,8 20,5
r2 25,20,2
2,6
25,20,3 r1 2,15
2,20
r3 30,20,7
minimum power assignment → [p1→r1 p2→r3 p3→r3] → 250 units of power
(c)

Example



p_1
2
p_2
2
p_3
2
p_4
5,10
5,10
5,10
5,10
r_1
2,5
r_2
5,10,2
5,10,2
2,5
2,5
2,5
2,6
r_4
2,6
r_3
5,10,3
5,10,2
minimum power assignment
$p_1 \to r_1$
$p_2 \to r_2$
$p_3 \to r_4$
$p_4 \to r_3$
77 units of power
(a)



p_1
2
p_2
2
p_3
2
p_4
5,10
5,10
5,10
5,10
r_1
2,6
r_2
5,10,2
5,10,2
2,5
2,5
2,5
2,6
r_4
2,6
r_3
5,10,3
5,10,2
minimum power assignment
$p_1 \to r_1$
$p_2 \to r_4$
$p_3 \to r_2$
$p_4 \to r_3$
75units of power
(b)

References

- Drozdowski, M. (2009). *Scheduling for parallel processing*. London: Springer.
- Sinnen, O. (2007). *Task scheduling for parallel systems* (Vol. 60). John Wiley & Sons.
- Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, *1*(1), 269-271.
- Andrade, H., Gedik, B., & Turaga, D. (2014). *Fundamentals of Stream Processing: Application Design, Systems, and Analytics*. Cambridge University Press.