THE UNIVERSITY of EDINBURGH
informatics
cisa | Centre for Intelligent Systems and their Applications

Regaining Control

Multi-Level Scheduling, Pilot-Jobs and a Fresh Perspective on Application Optimisation

13/04/2012 | ole.weidner@ed.ac.uk

Outline

THE UNIVERSITY of EDINBURGH
informatics
cisa | Centre for Intelligent Systems and their Applications

Background and Context

Part I
1. Multi-Level Scheduling
2. Pilot-Job Systems
3. Application Use-Cases

Part II
4. A Case for Optimization
5. Application Optimization on Pilot-Job Level
6. Integration with a Pilot-Job System

Background and Context

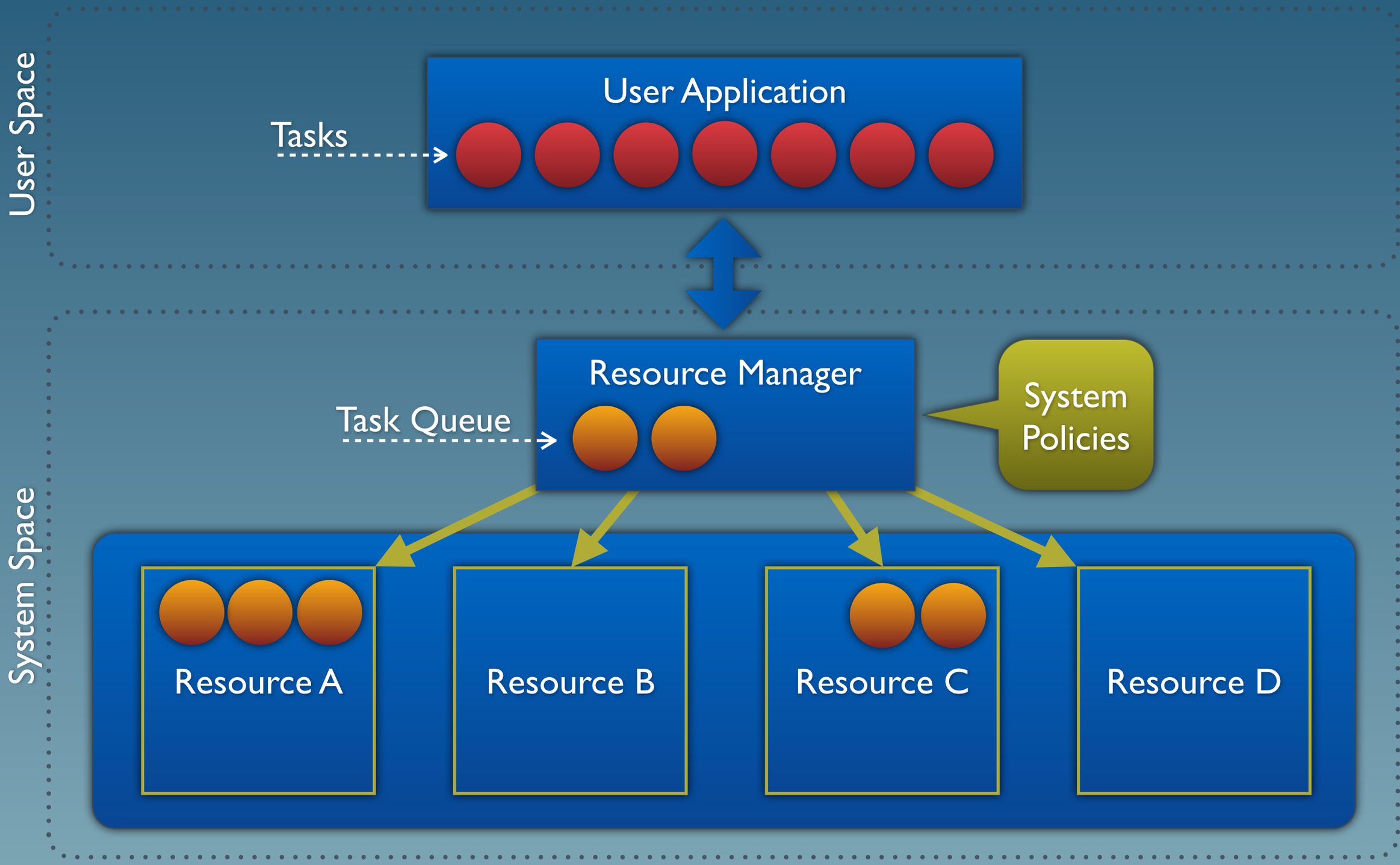
THE UNIVERSITY of EDINBURGH
informatics
cisa | Centre for Intelligent Systems and their Applications

- Ole Weidner
  - Research IT Manager at Louisiana State University
    - Soon to be found at Rutgers University, New Jersey
    - Technical lead of the SAGA project
  - Part-time PhD student at University of Edinburgh
    - Loosely involved with some projects in the DIR group
- Research Interests
  - Distributed systems and e-Infrastructure development
  - Methods, strategies and frameworks to support dynamic and data-intensive applications

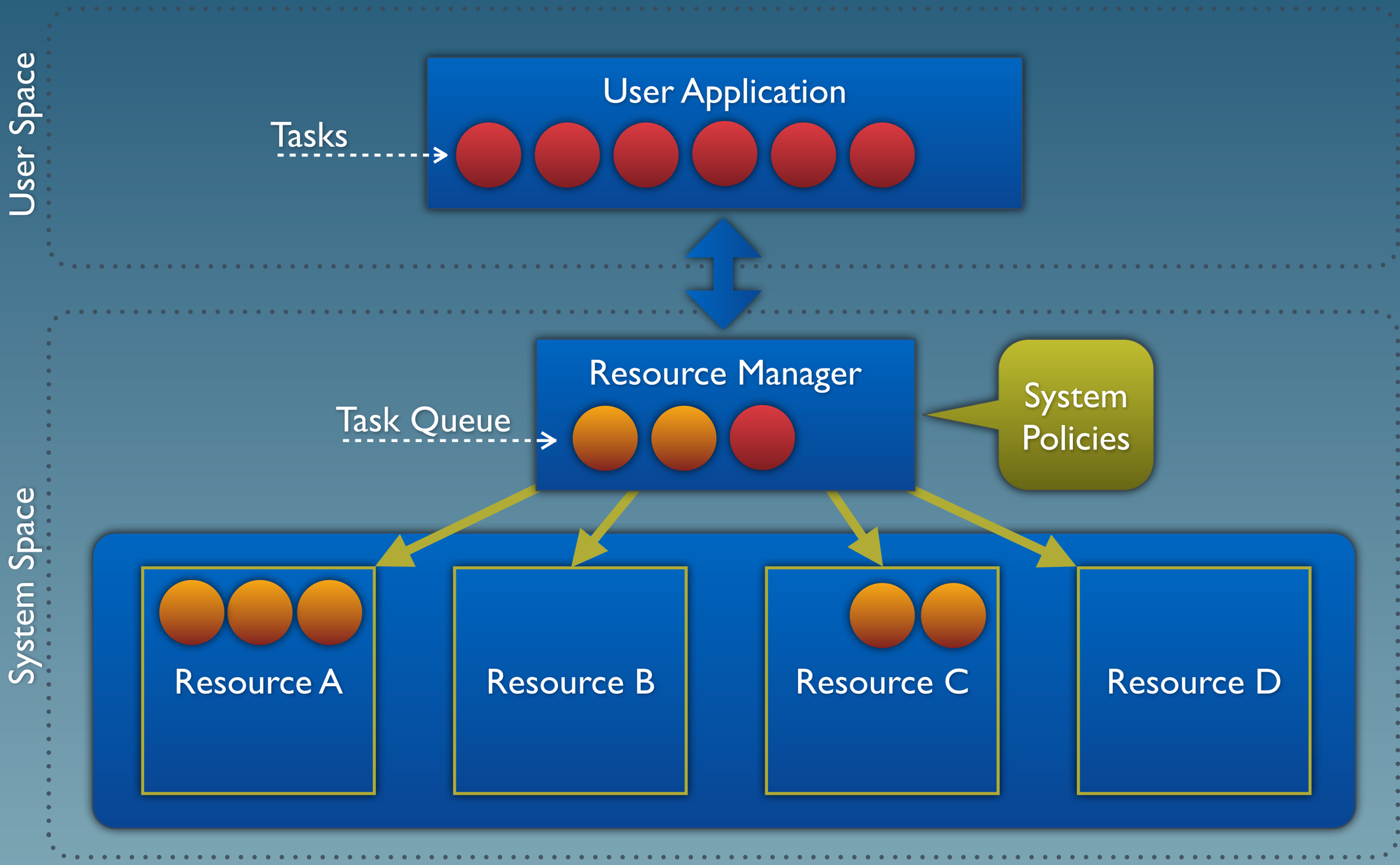
1. Multi-Level Scheduling

- In distributed computing, MLS is a provisioning technique that allows user-level resource managers to control resources that are normally controlled by system-level resource managers
  - Resource providers 'lease' resources to the user
  - User is responsible for task scheduling and management
  - = Logical separation between provisioning and usage
- This is different from the 'common' usage mode where the user / the application submits jobs directly to the resource provider (e.g., an HPC scheduler) which then takes care of scheduling and management

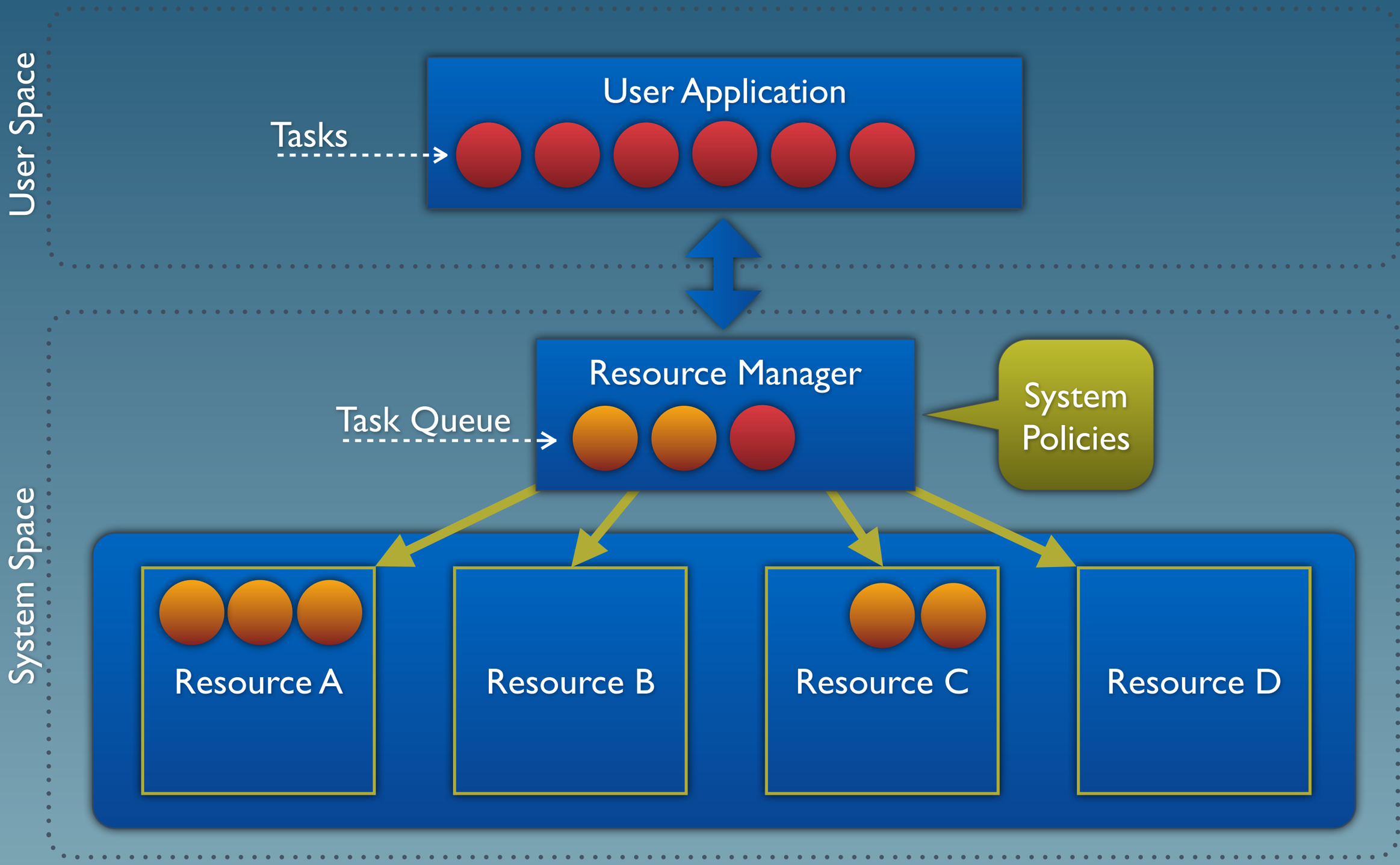
1. Multi-Level Scheduling



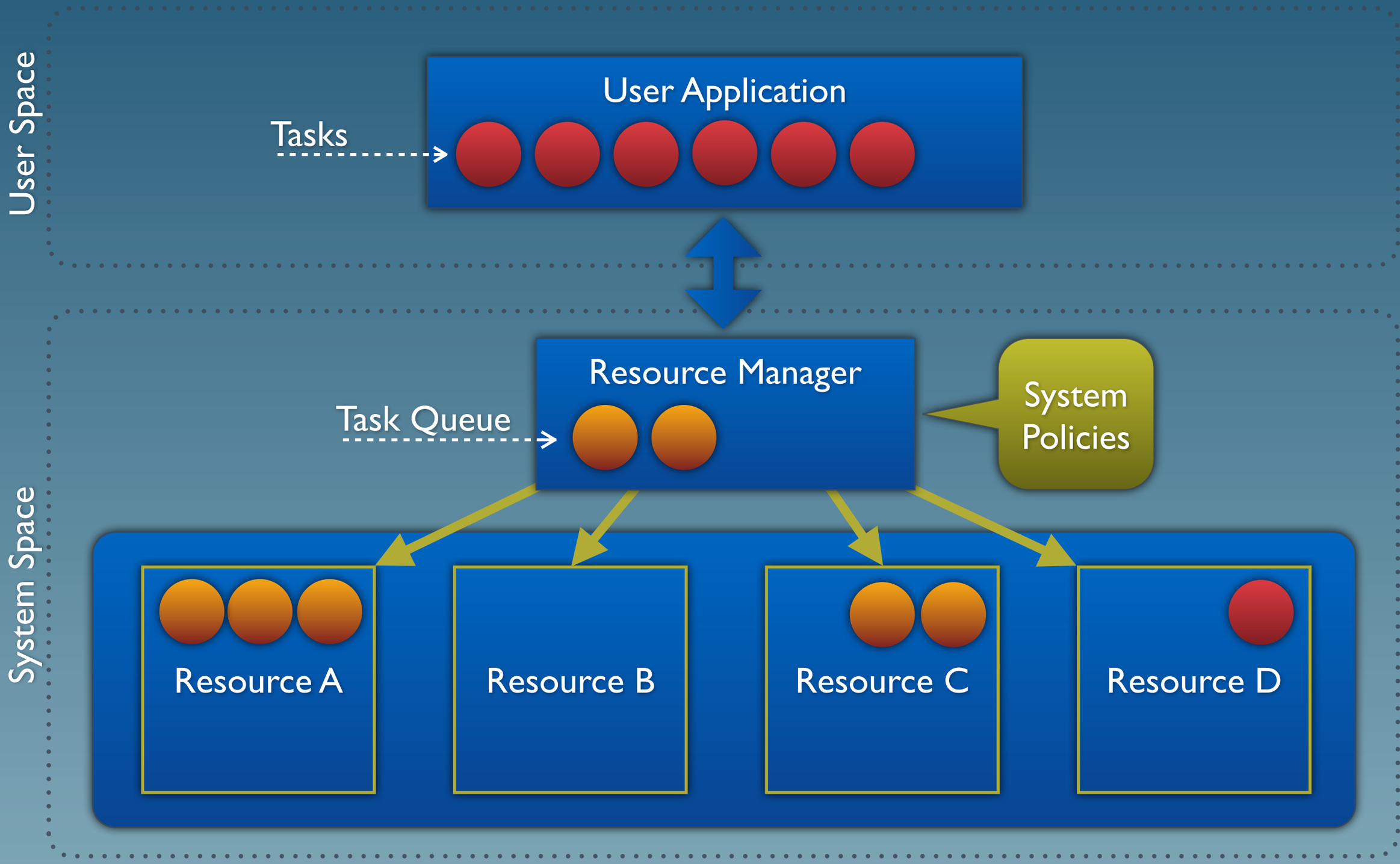
1. Multi-Level Scheduling



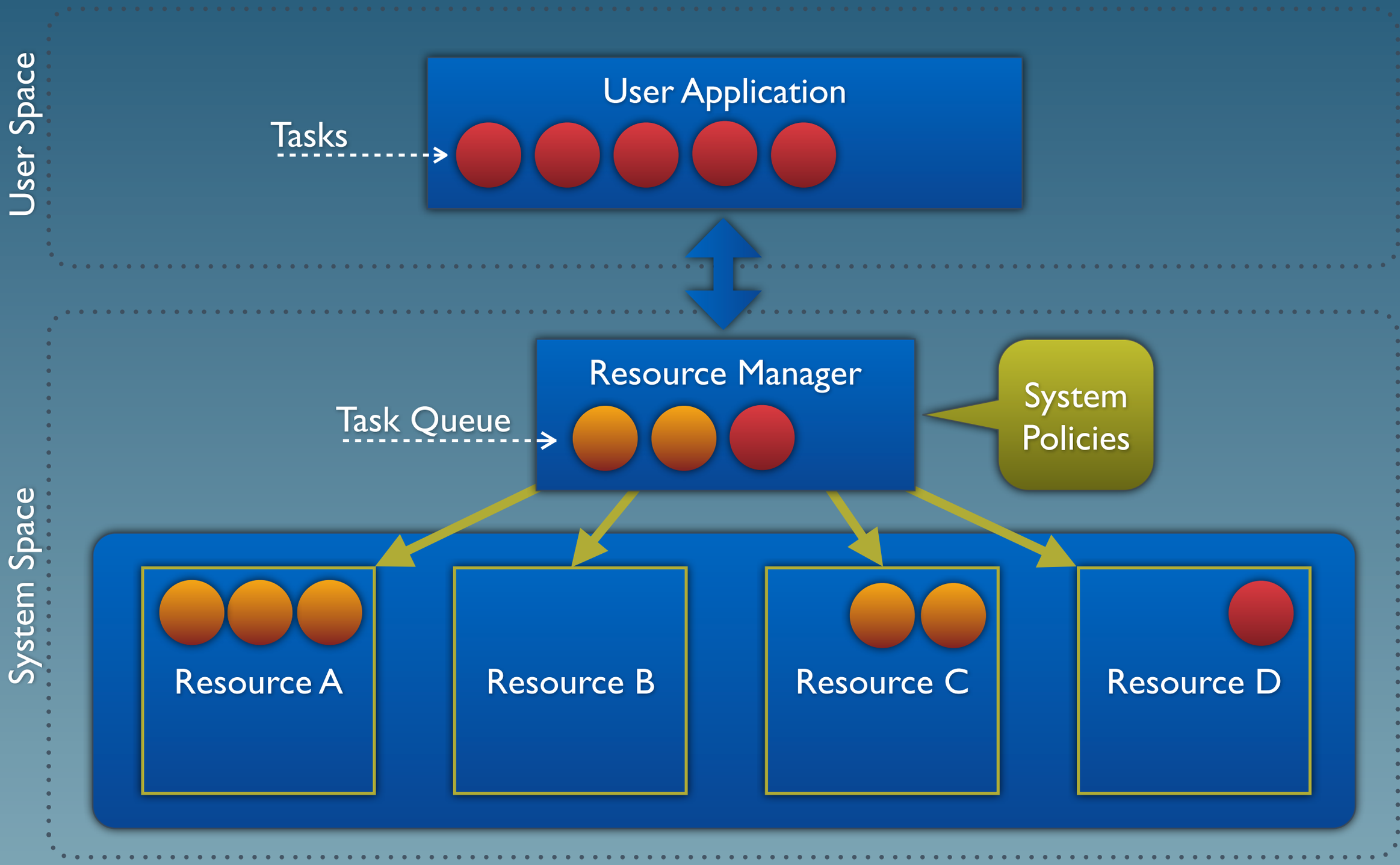
1. Multi-Level Scheduling



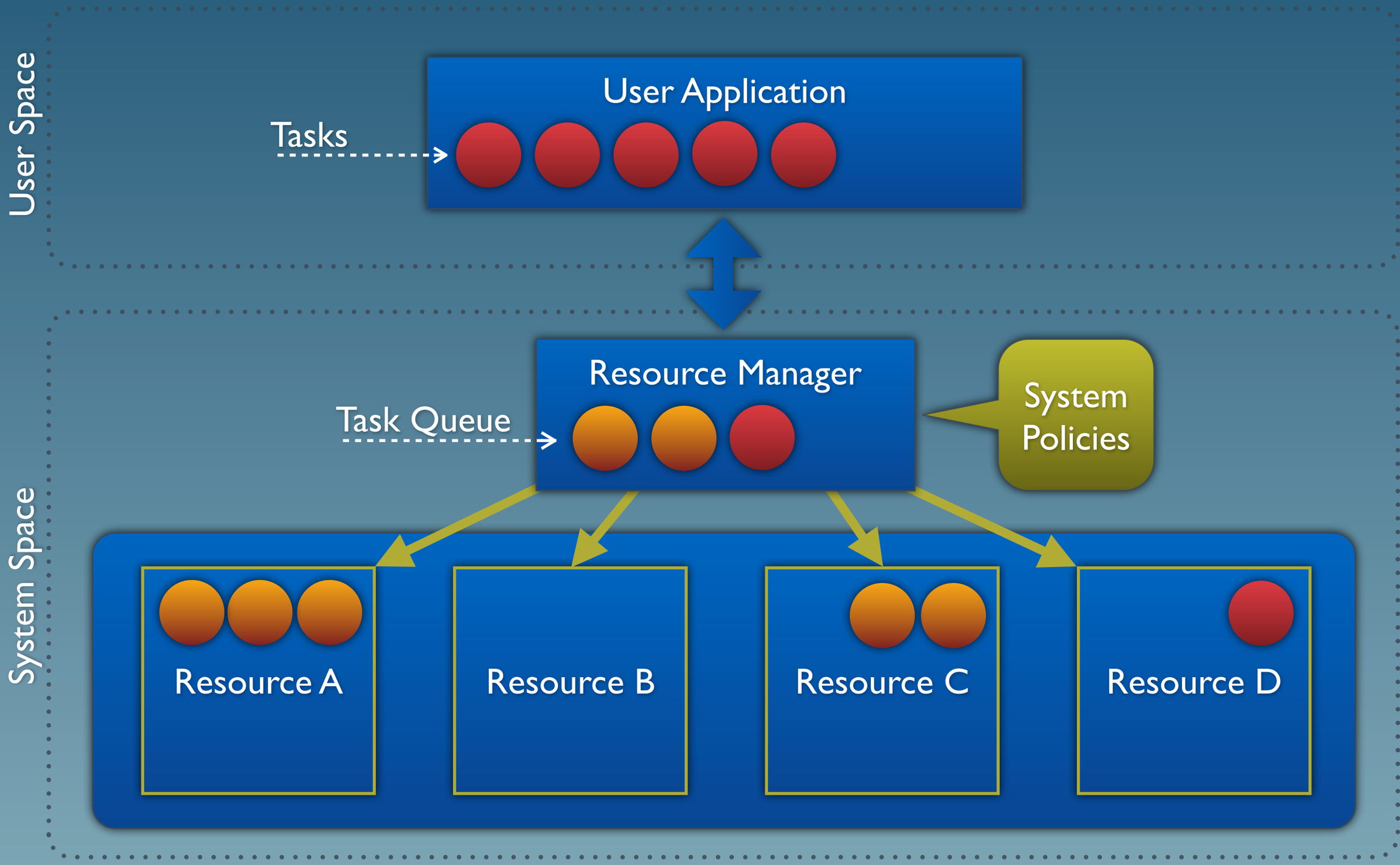
1. Multi-Level Scheduling



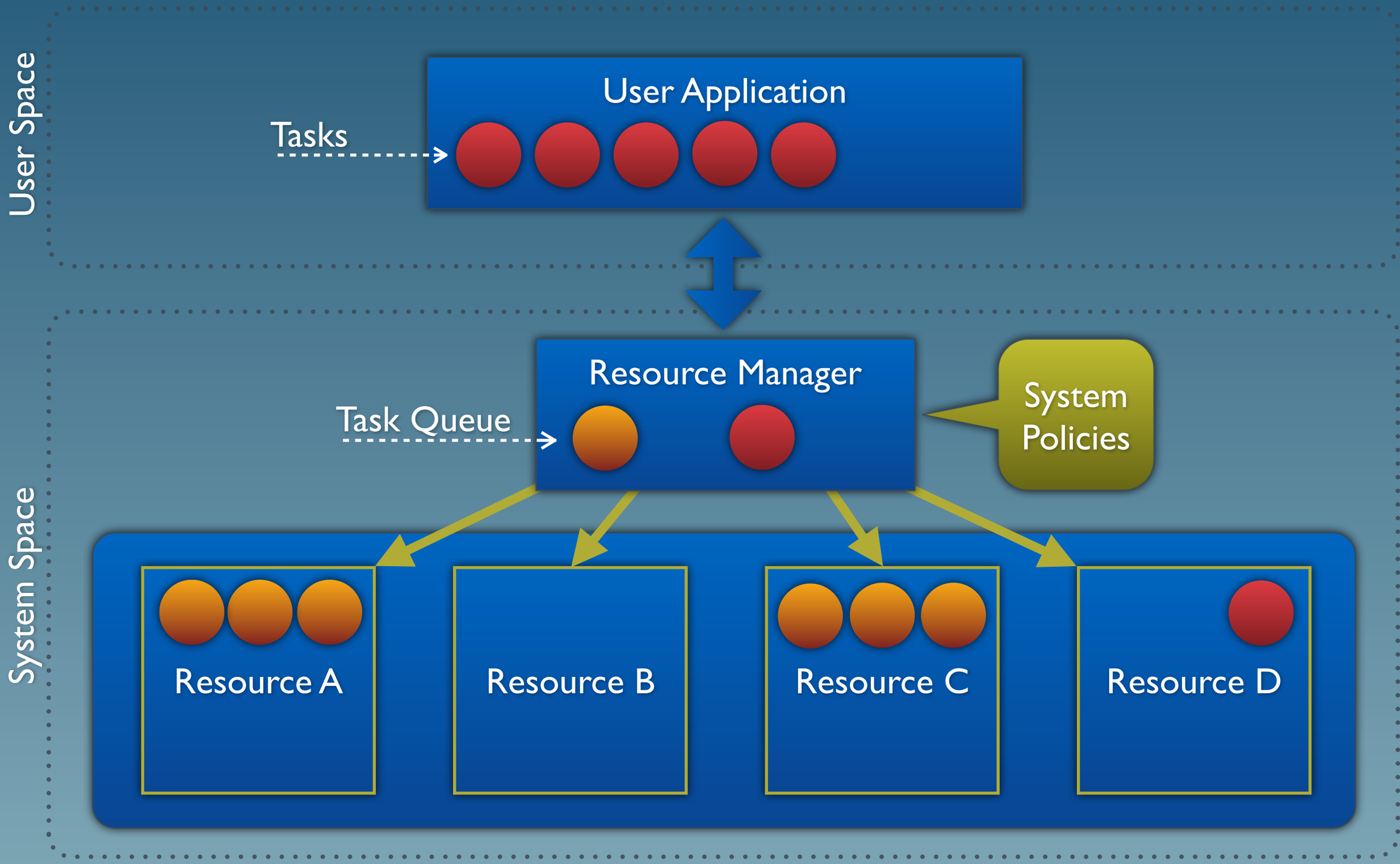
1. Multi-Level Scheduling



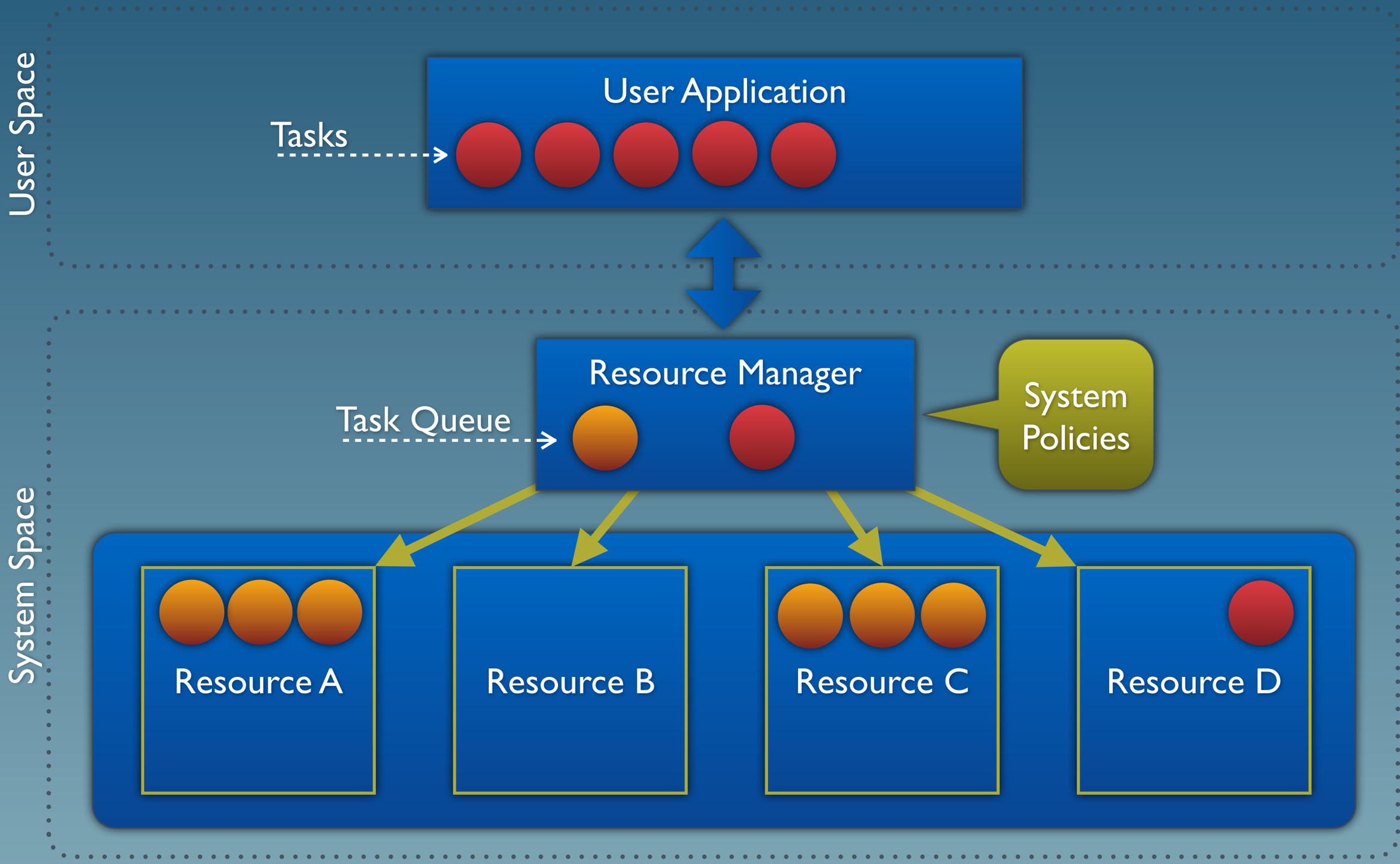
1. Multi-Level Scheduling



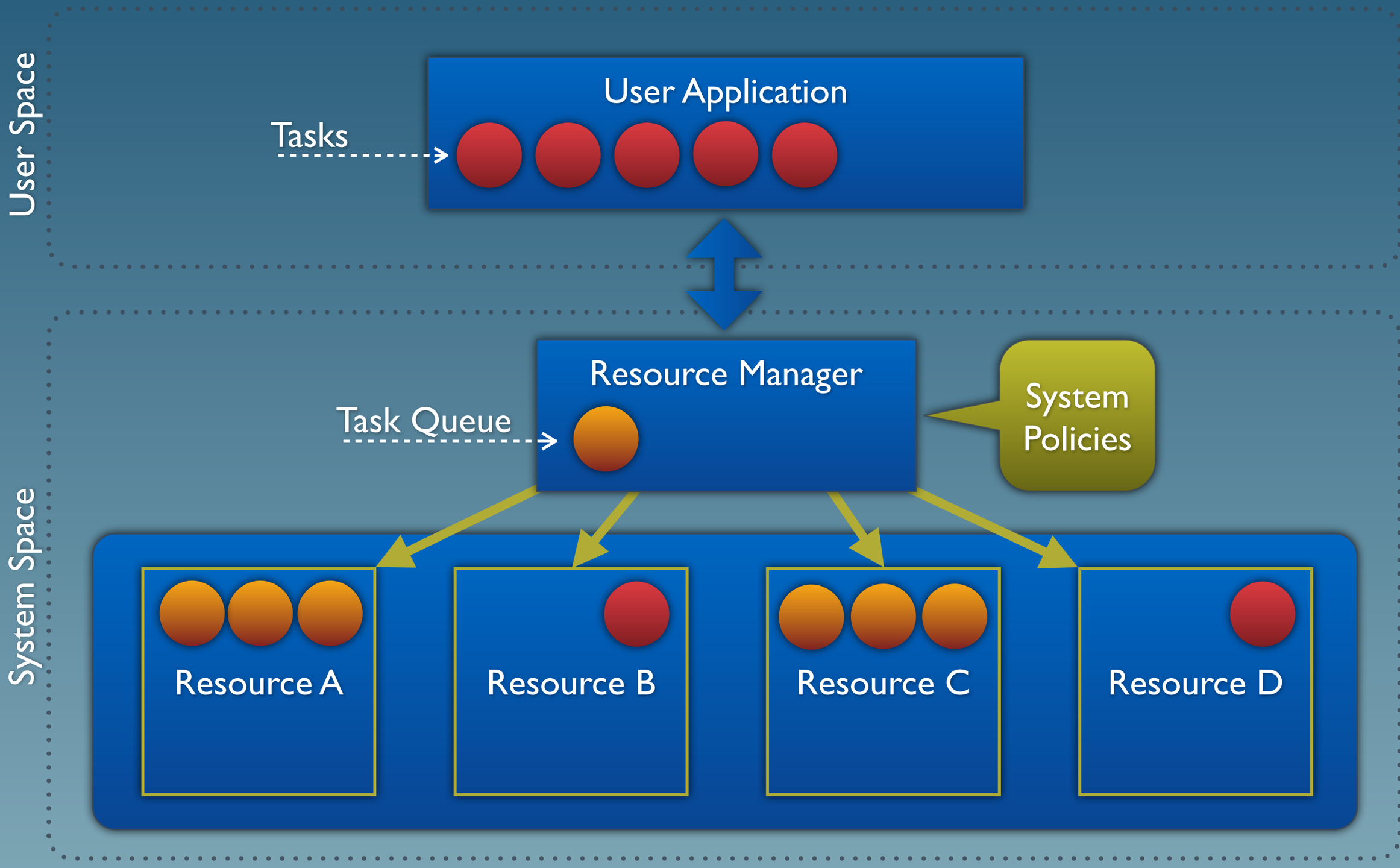
1. Multi-Level Scheduling



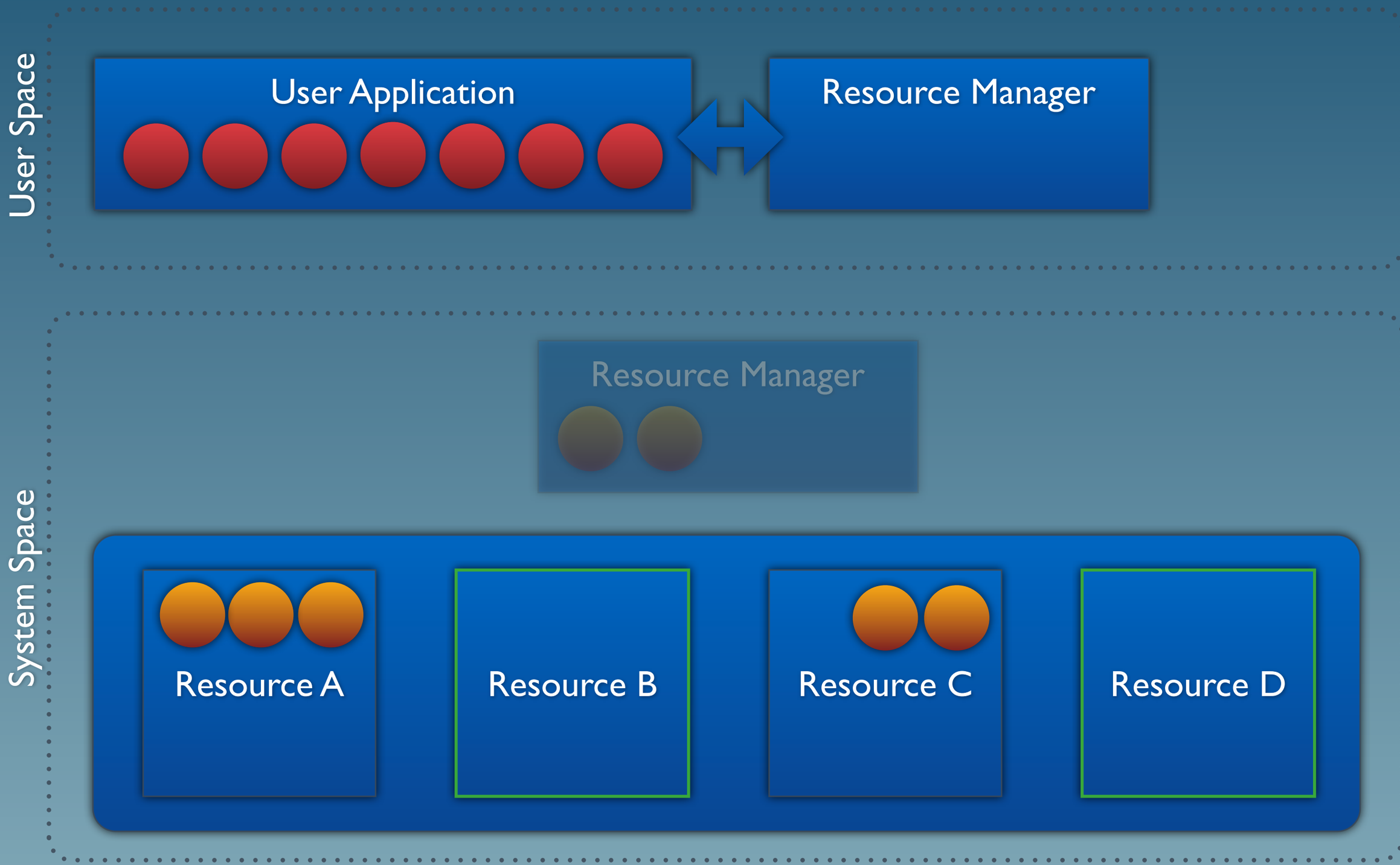
1. Multi-Level Scheduling



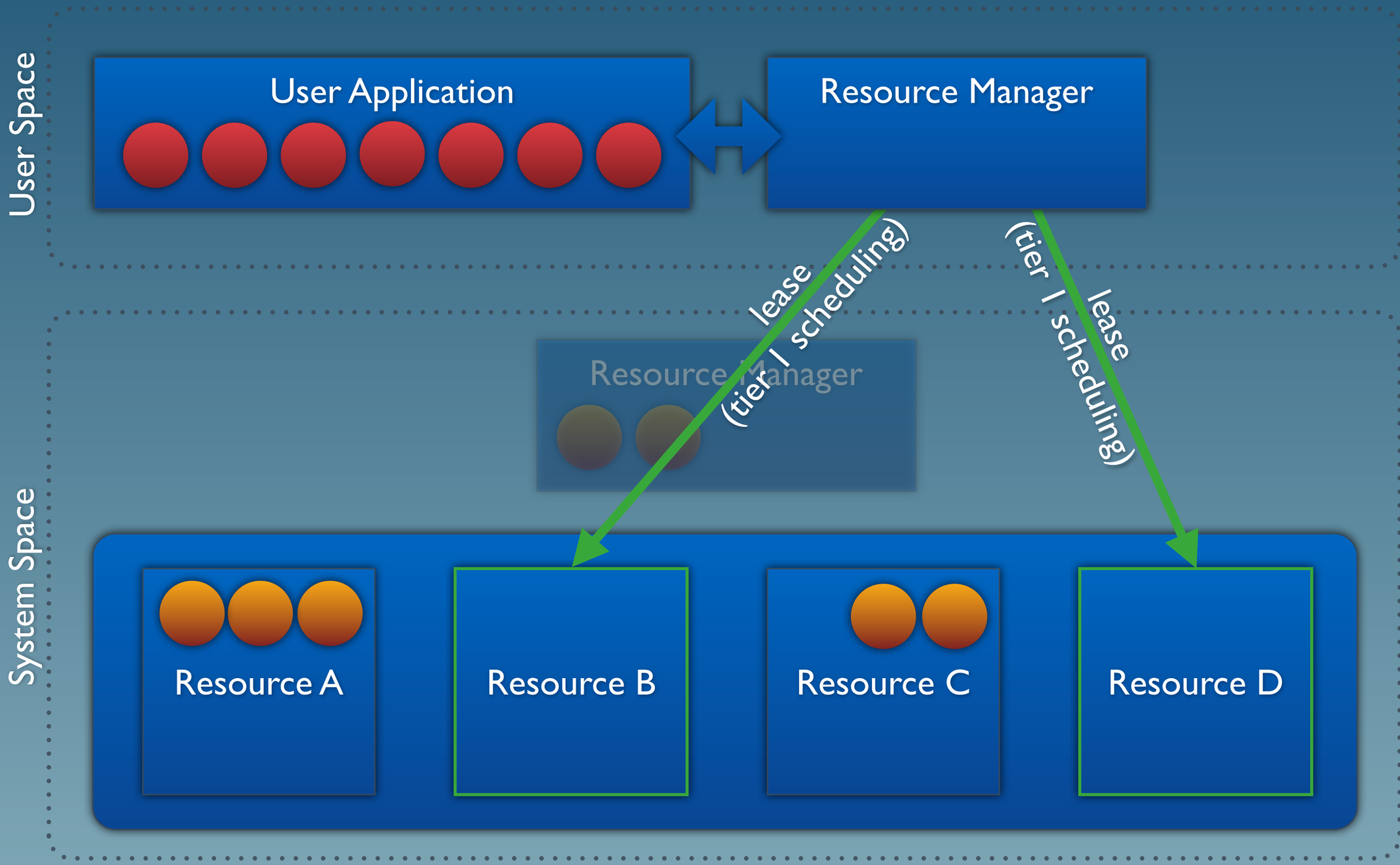
1. Multi-Level Scheduling



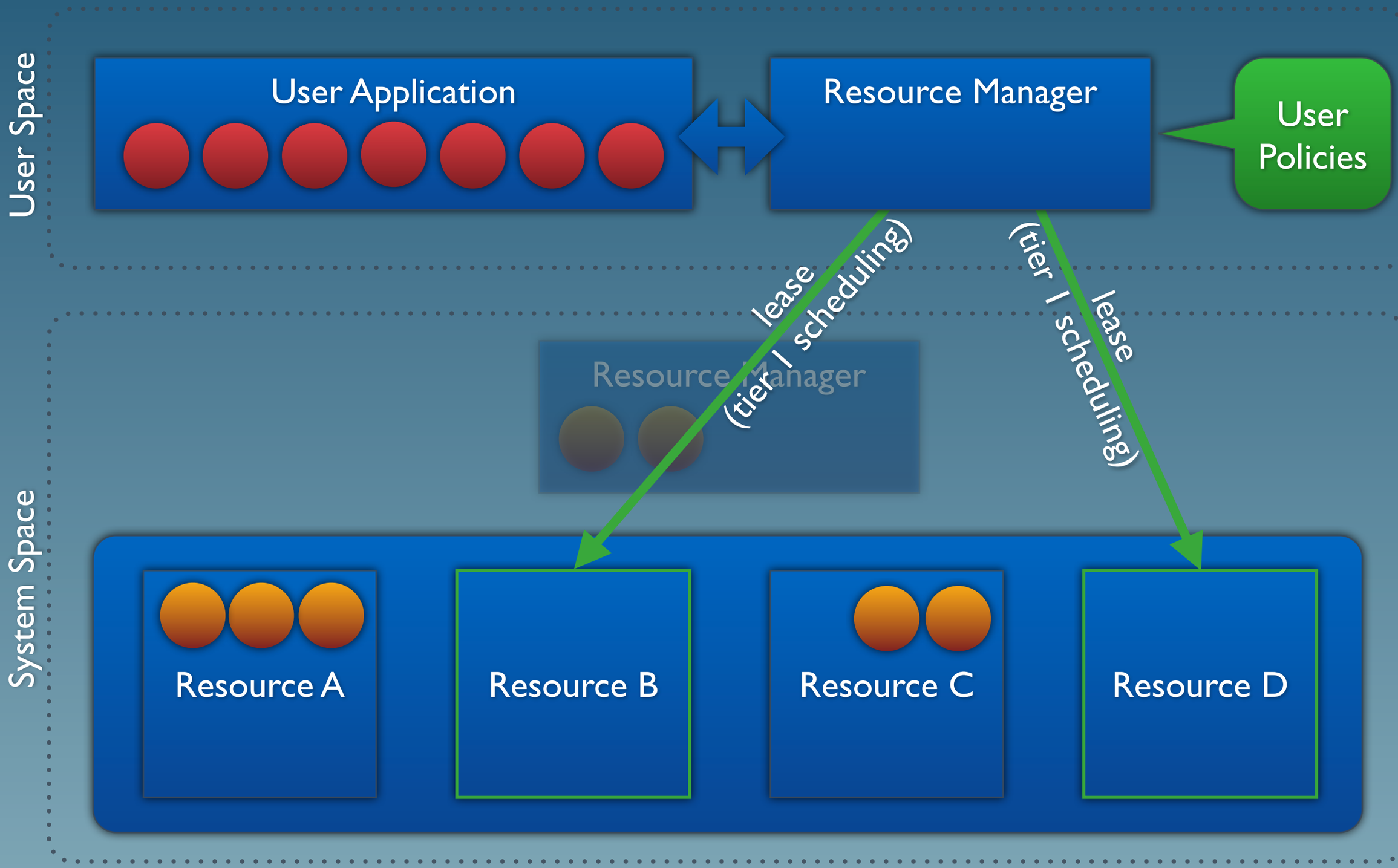
1. Multi-Level Scheduling



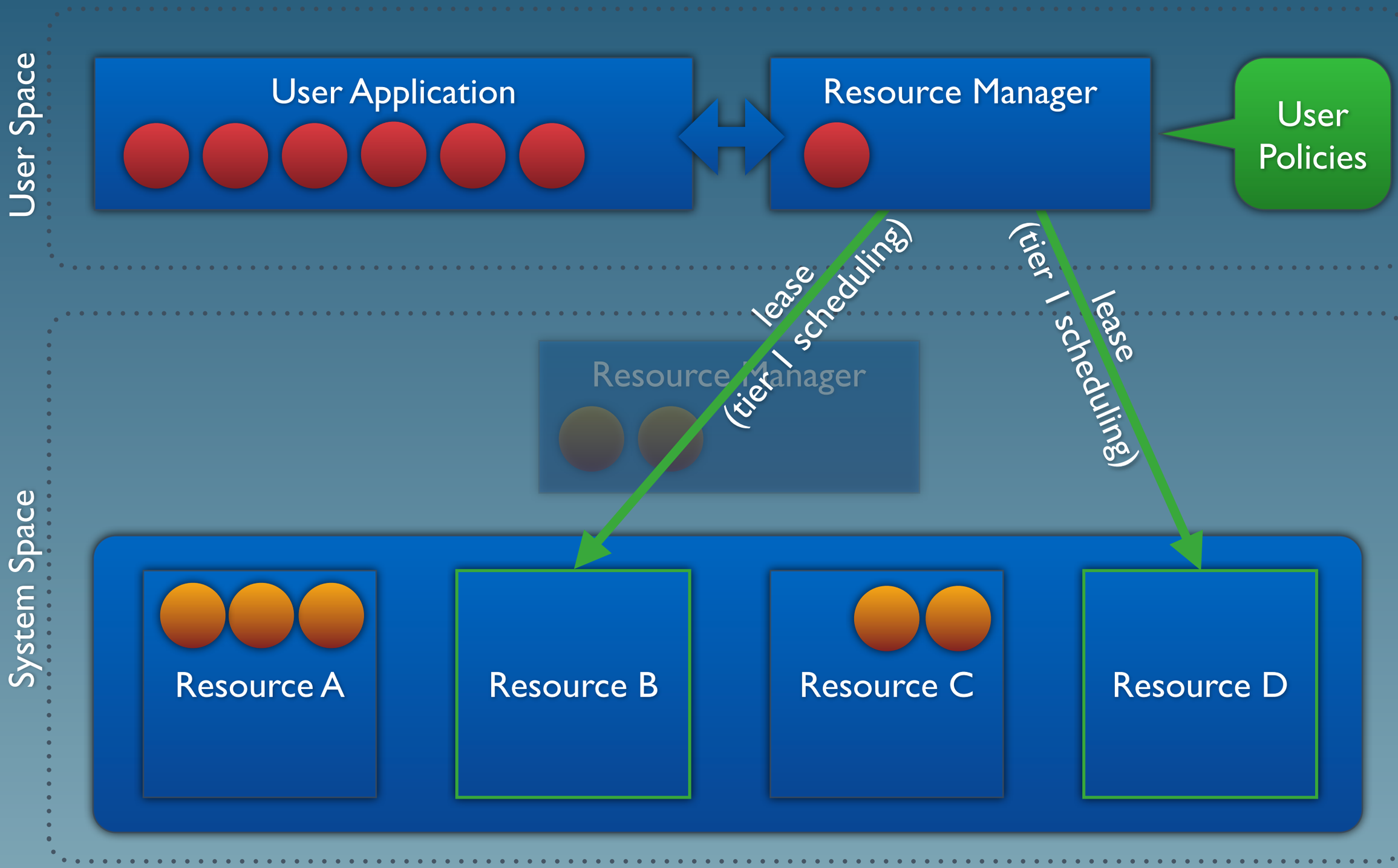
1. Multi-Level Scheduling



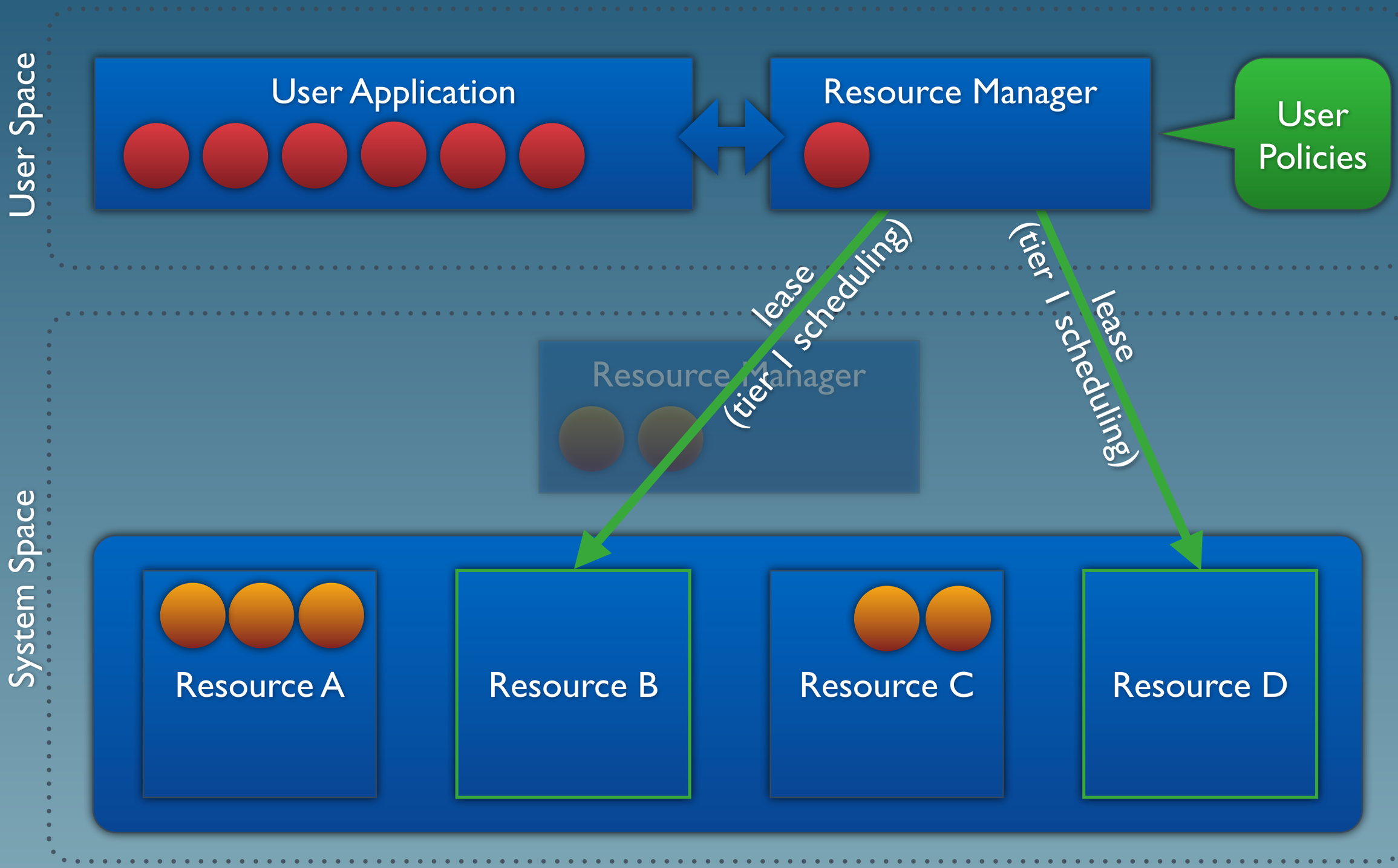
1. Multi-Level Scheduling



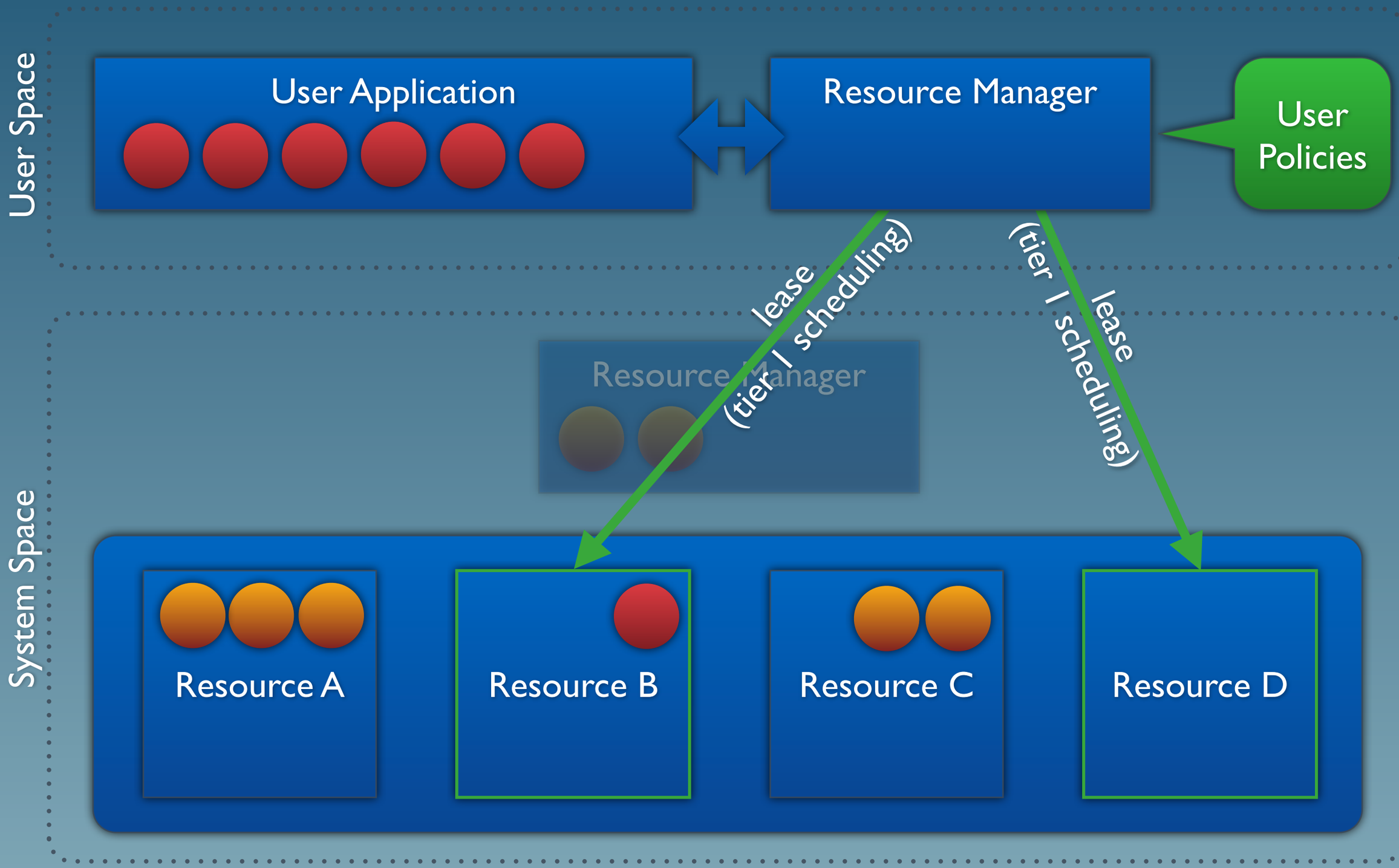
1. Multi-Level Scheduling



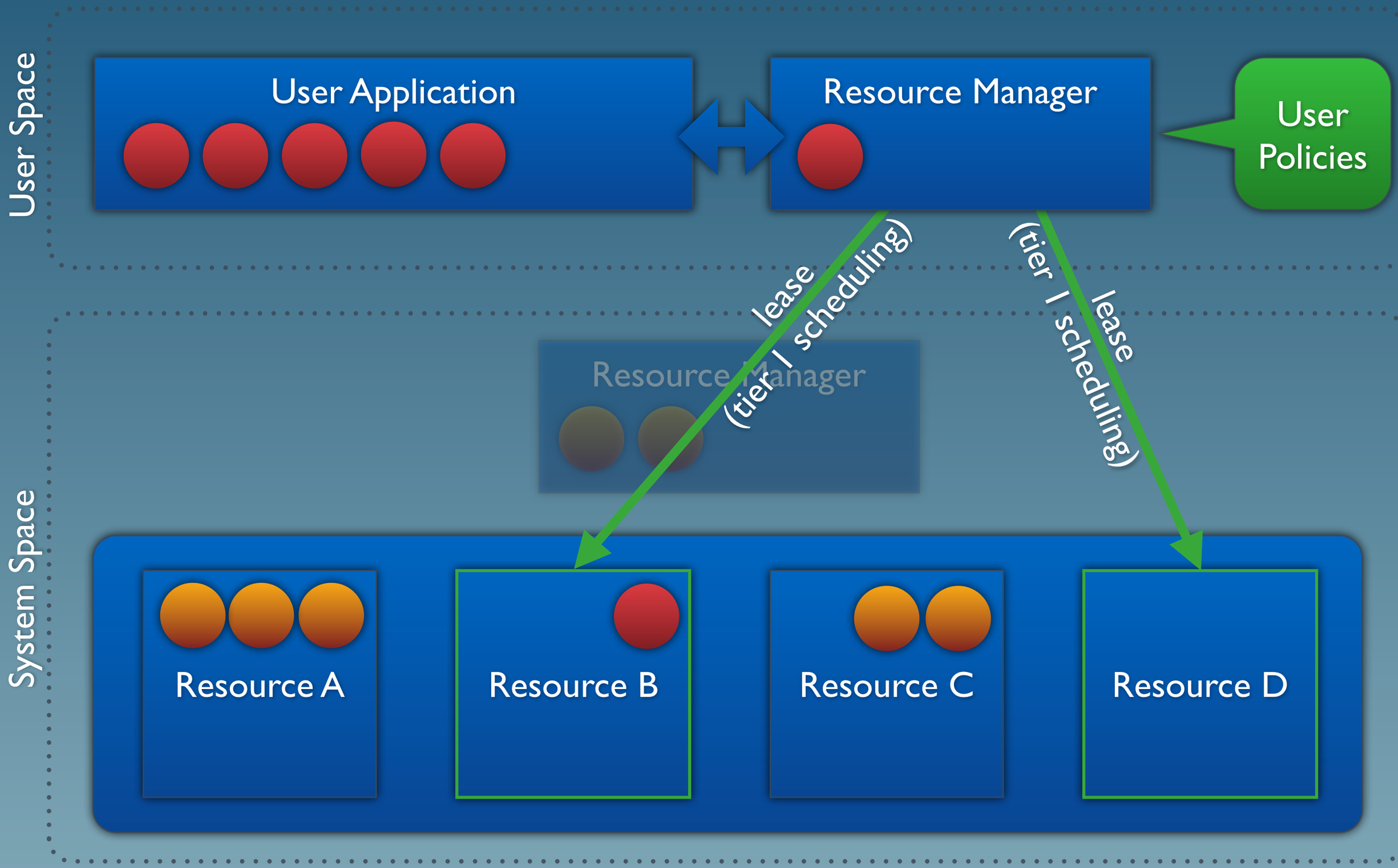
1. Multi-Level Scheduling



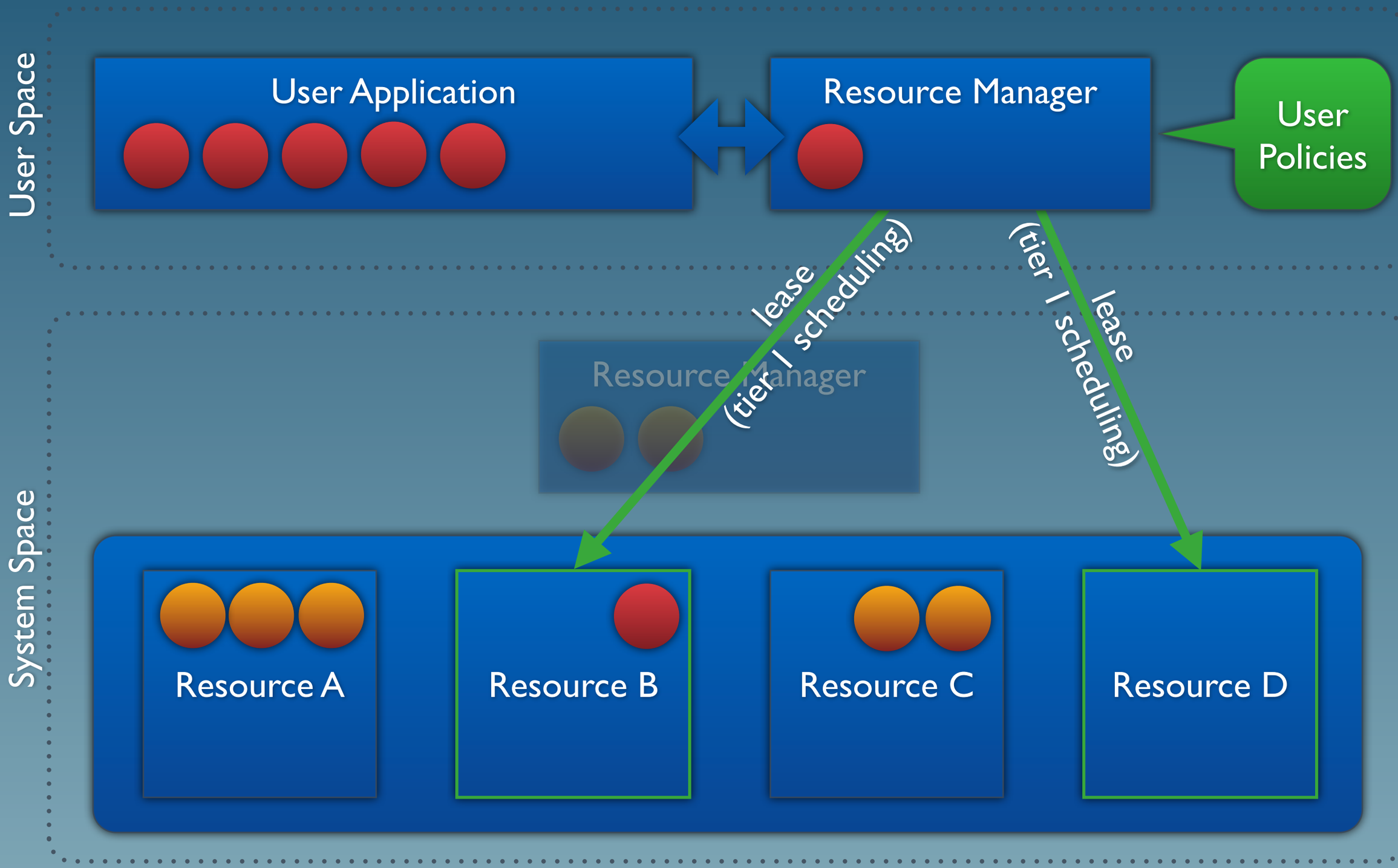
1. Multi-Level Scheduling



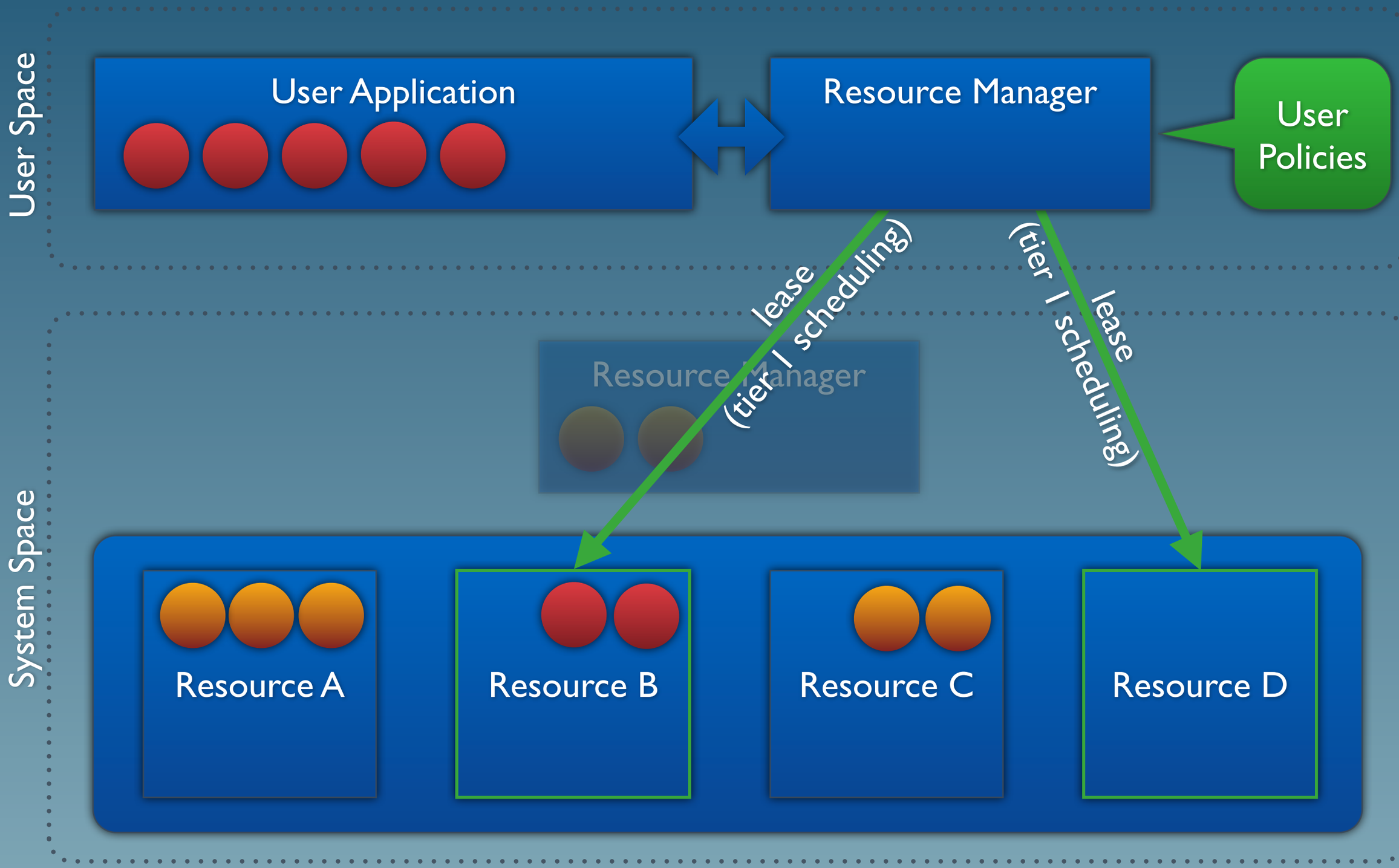
1. Multi-Level Scheduling



1. Multi-Level Scheduling



1. Multi-Level Scheduling



1. Multi-Level Scheduling

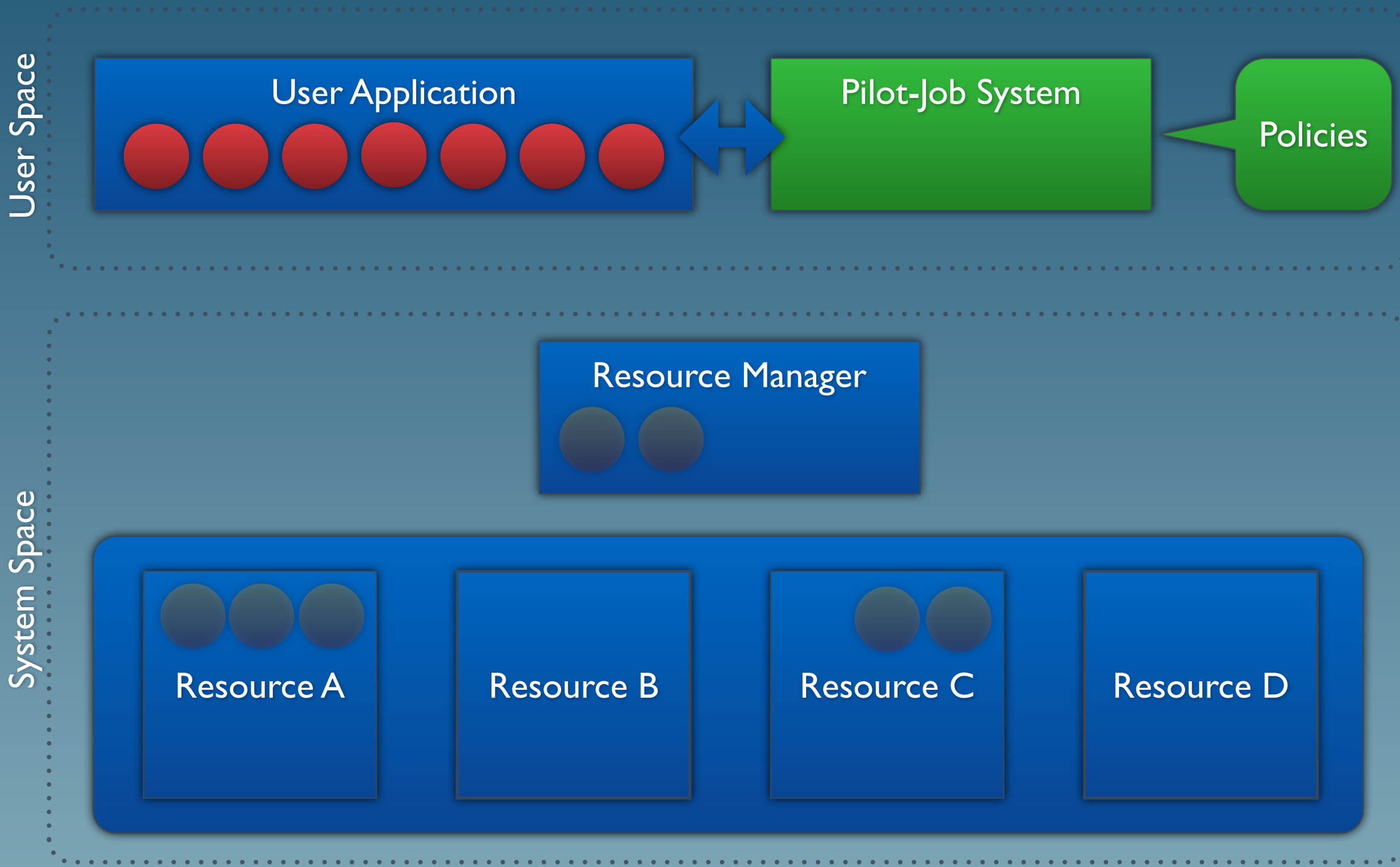
- Multi-level scheduling is a common pattern. Similar concepts can be found, e.g., in:
  - Infrastructure as a Service (IaaS)
  - OS thread scheduling
- Why Multi-Level Scheduling? Can't we just leave everything to the (system-level) resource managers?
- Answer: it depends!
  - Can your application be executed efficiently without control over resource manager policies, like task & data placement, job type preferences or job quantity limits?
  - Is system-level resource management overhead a concern?

2. Pilot-Job Systems

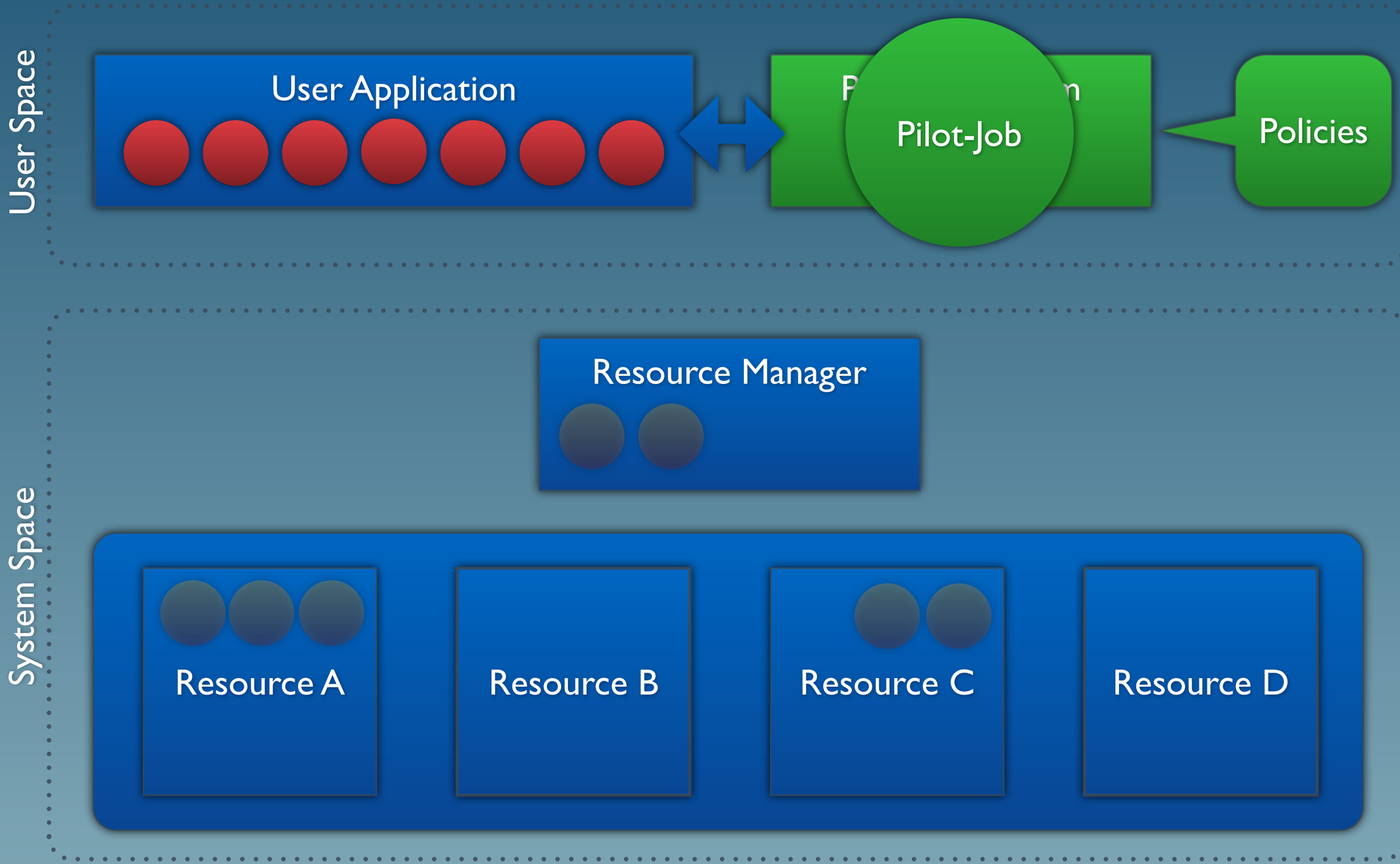
- What are Pilot-Job Systems?
  - A "best practice" implementation strategy for multi-level scheduling on HPC resources (but also others)
  - PJS realise resource "leasing" via agents ("pilot-jobs") that are submitted via the system-level resource managers
  - Once pilot-jobs become active, they become the resources (resource overlay) for the user-level (pilot-job system) resource manager
  - Application tasks are submitted to the user-level resource manager which then schedules them to available pilot-job resources according to its policies

2. Pilot-Job Systems

THE UNIVERSITY of EDINBURGH
informatics
cisa | Centre for Intelligent Systems and their Applications

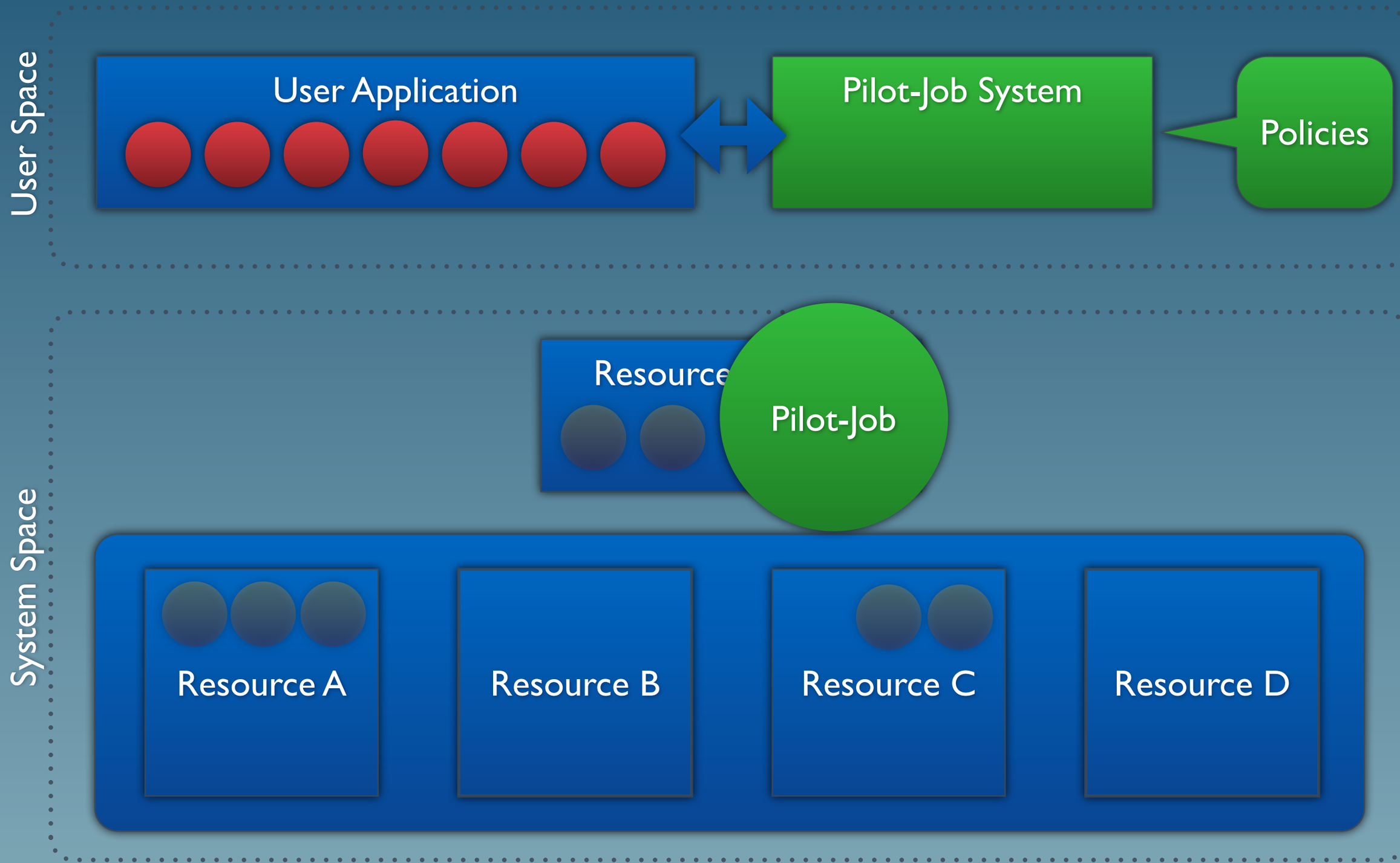


2. Pilot-Job Systems

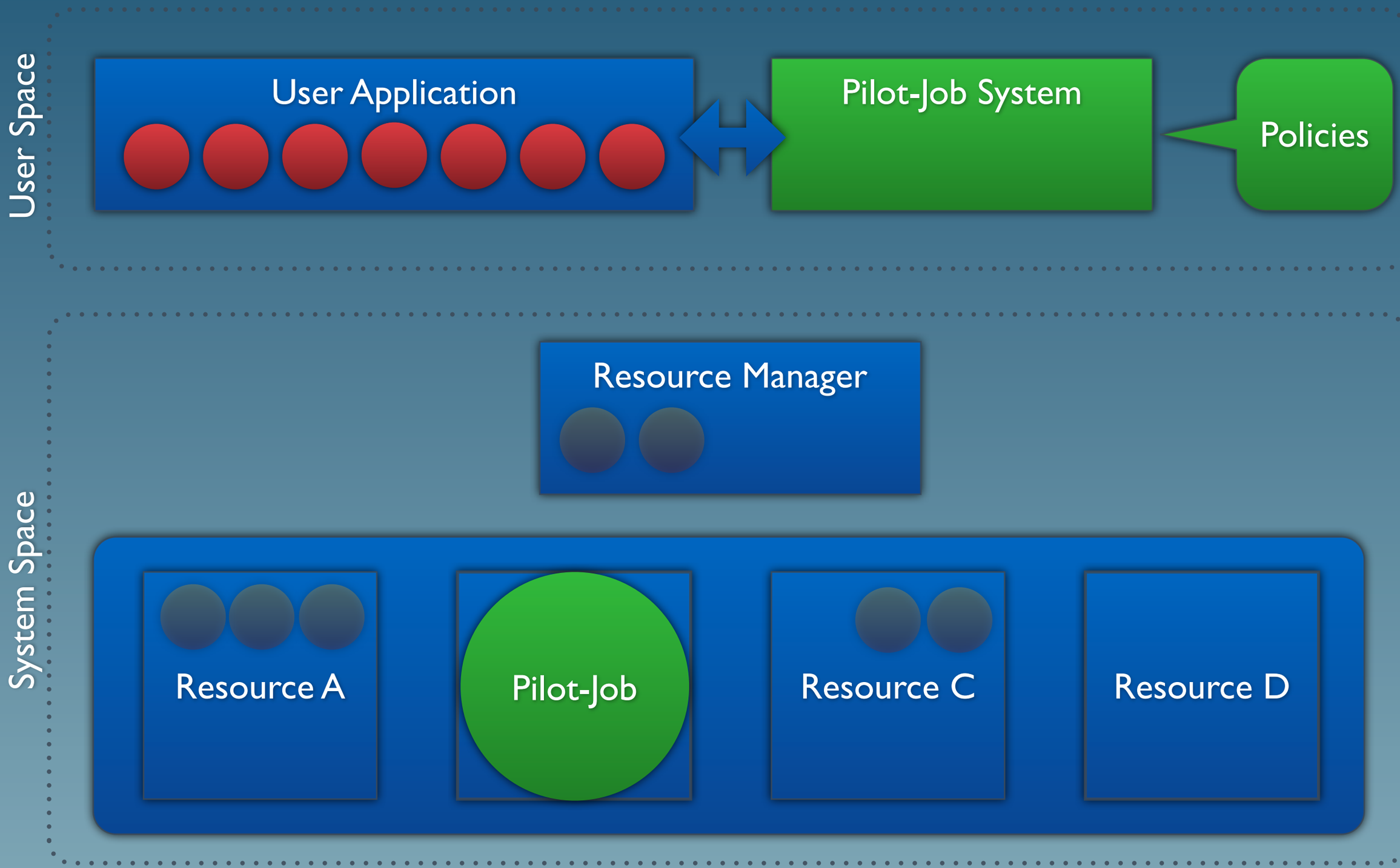


2. Pilot-Job Systems

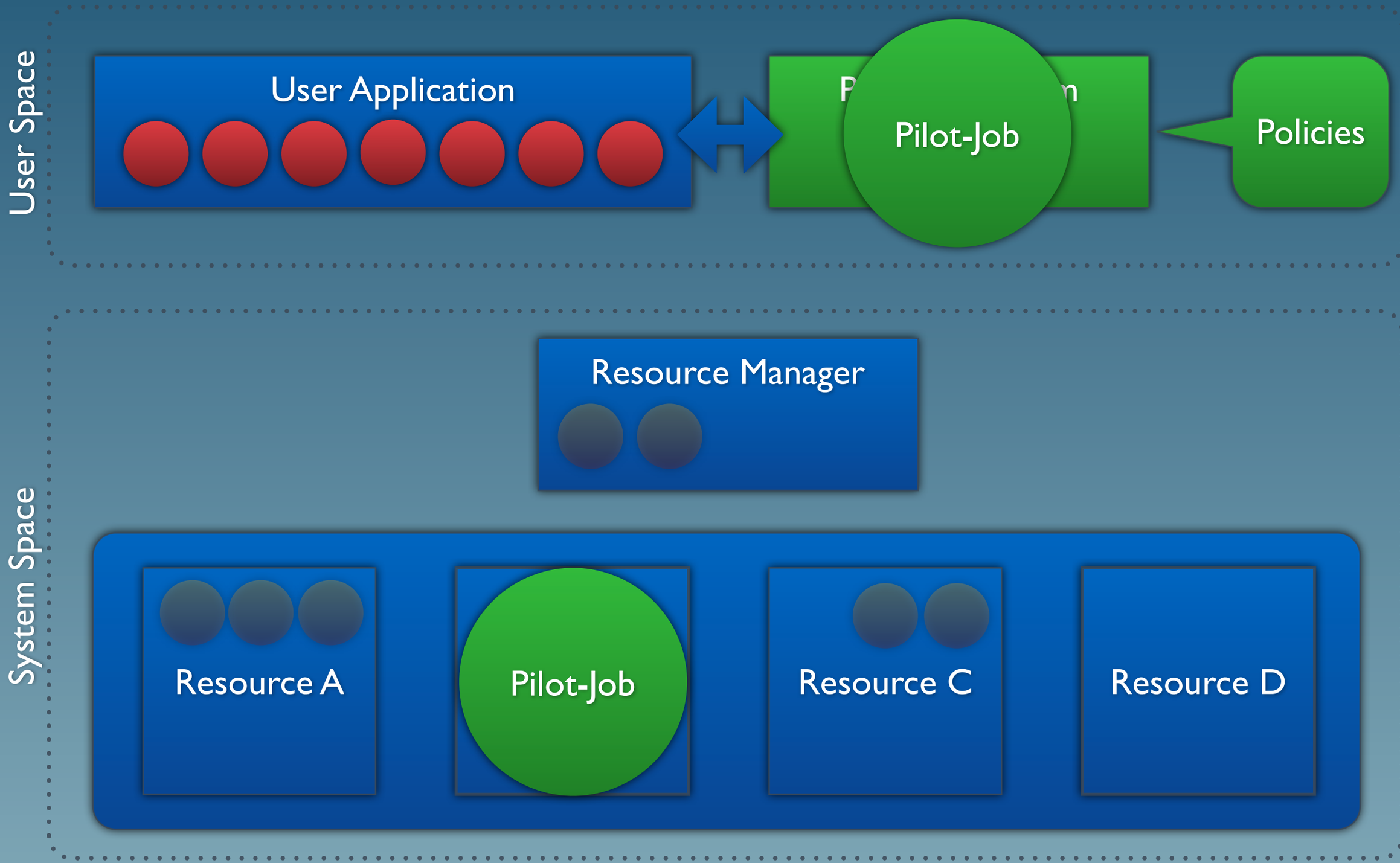
THE UNIVERSITY of EDINBURGH
informatics
cisa | Centre for Intelligent Systems and their Applications



2. Pilot-Job Systems

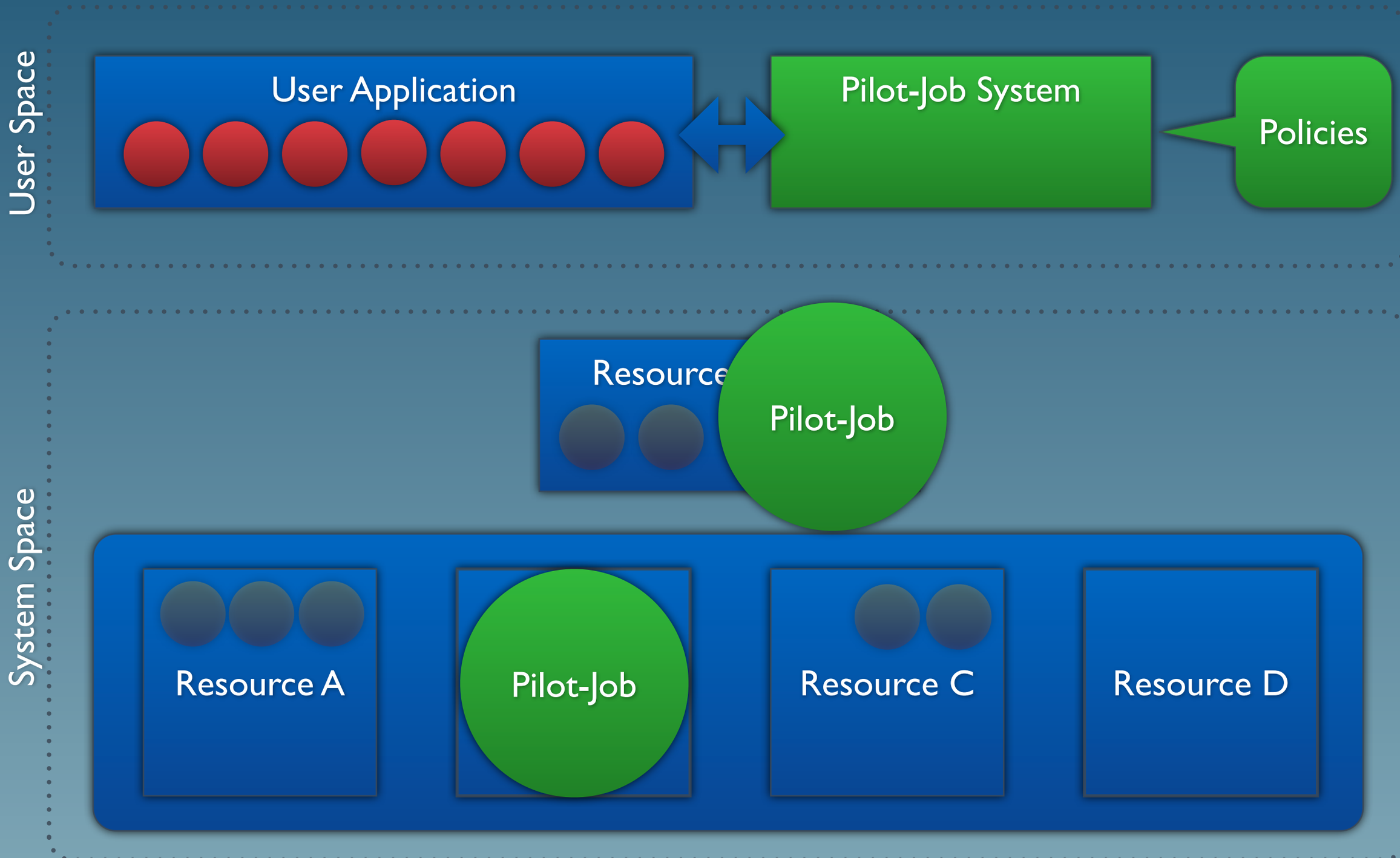


2. Pilot-Job Systems



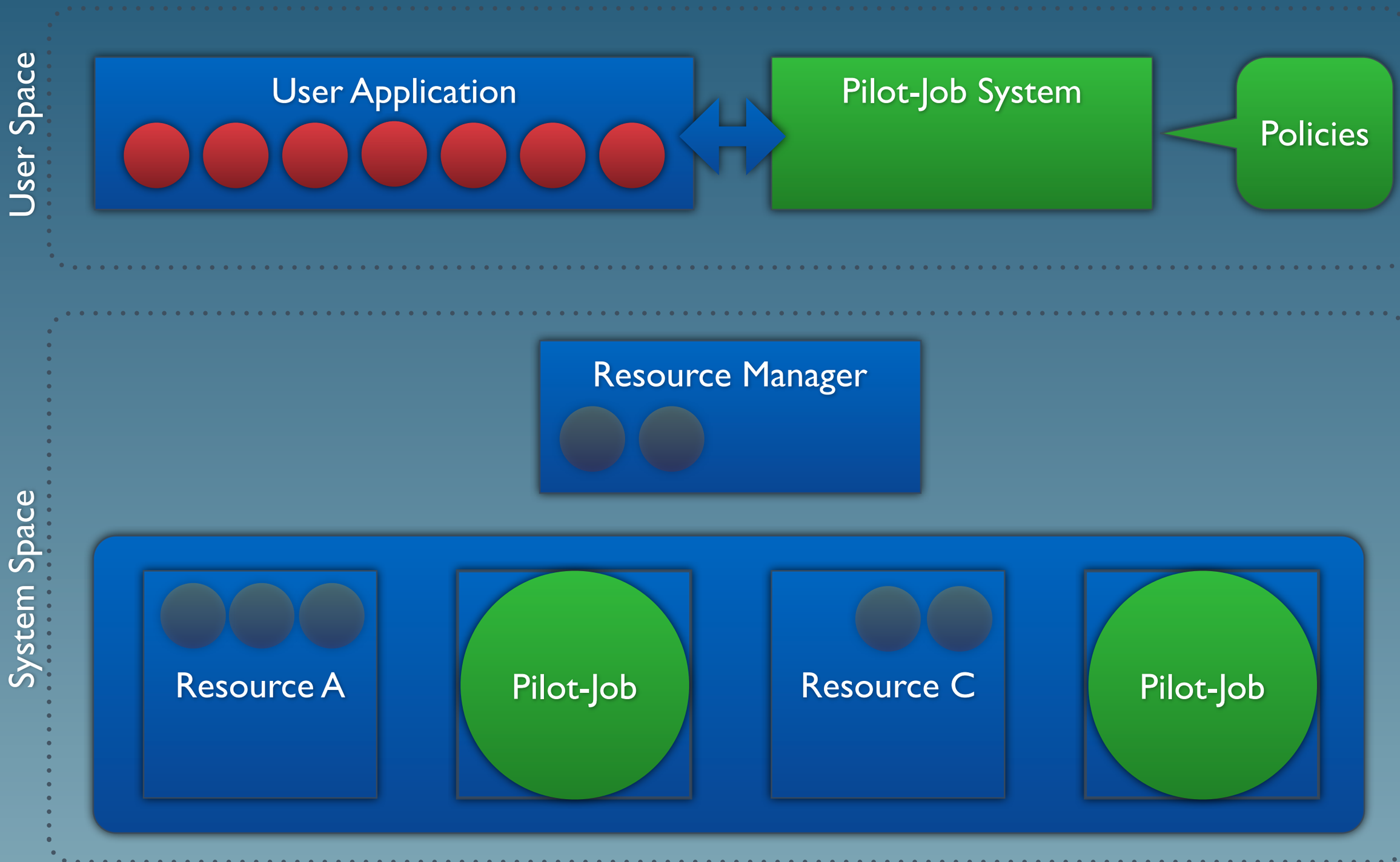
2. Pilot-Job Systems

THE UNIVERSITY of EDINBURGH
informatics
cisa | Centre for Intelligent Systems and their Applications

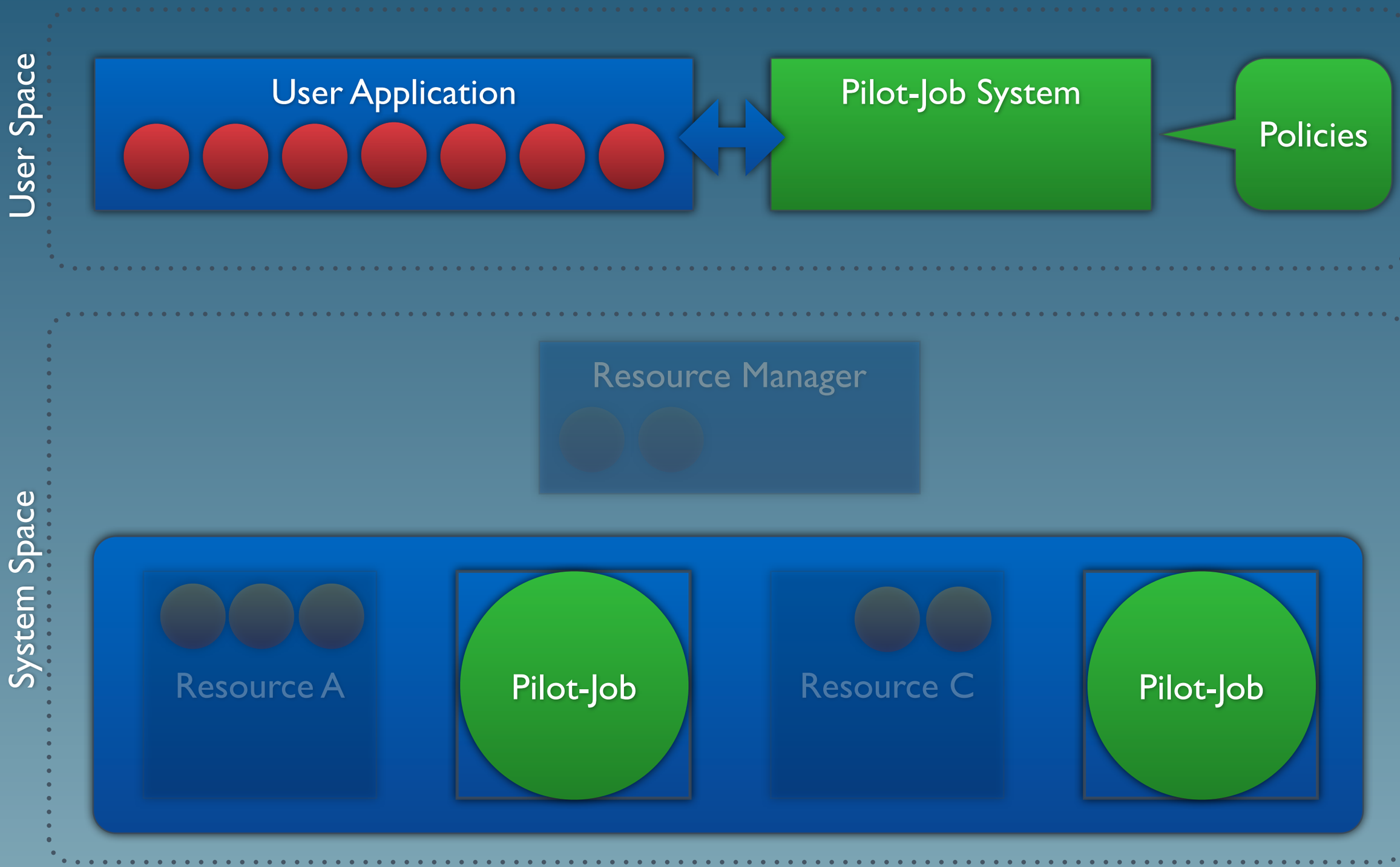


2. Pilot-Job Systems

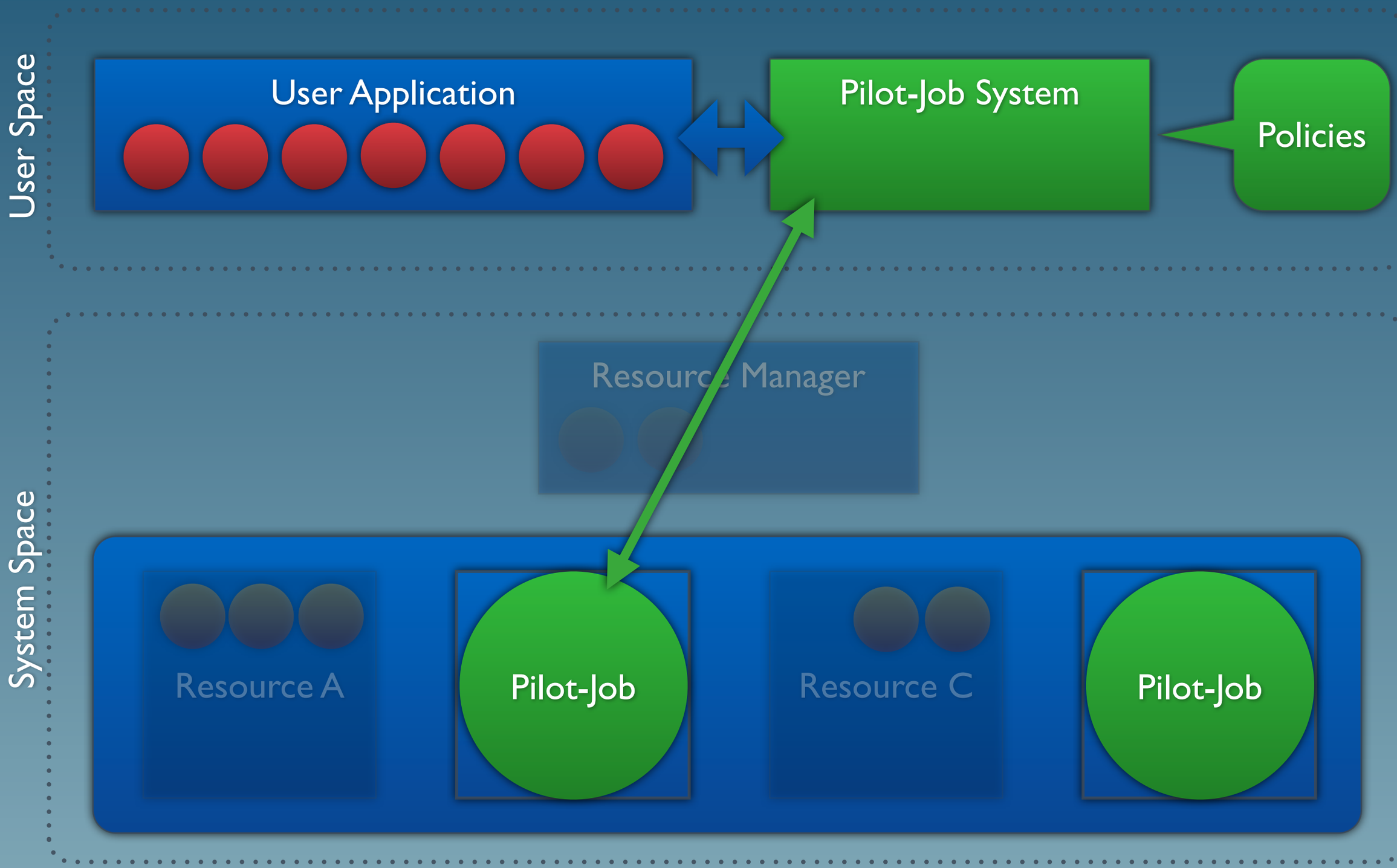
THE UNIVERSITY of EDINBURGH
informatics
cisa | Centre for Intelligent Systems and their Applications



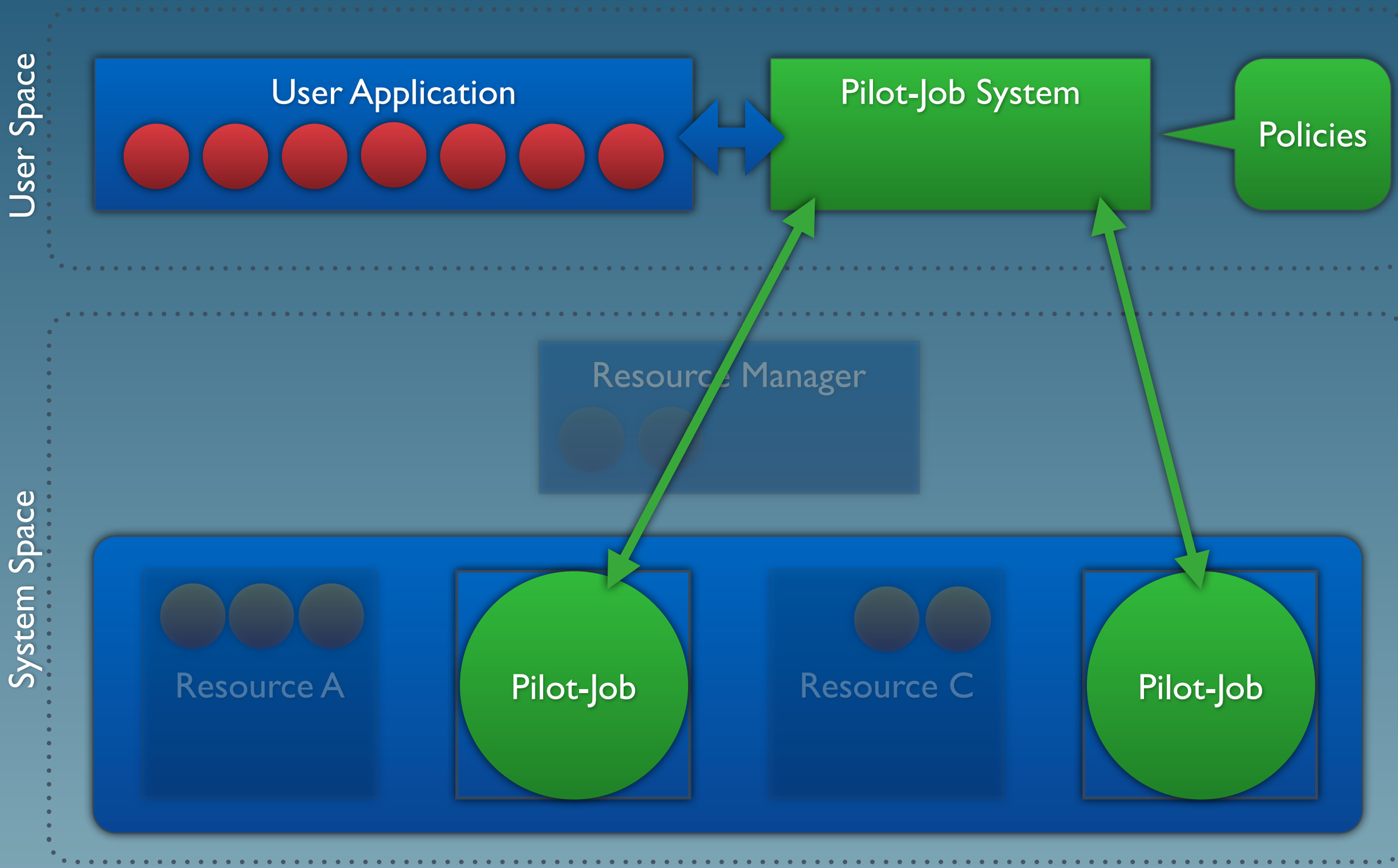
2. Pilot-Job Systems



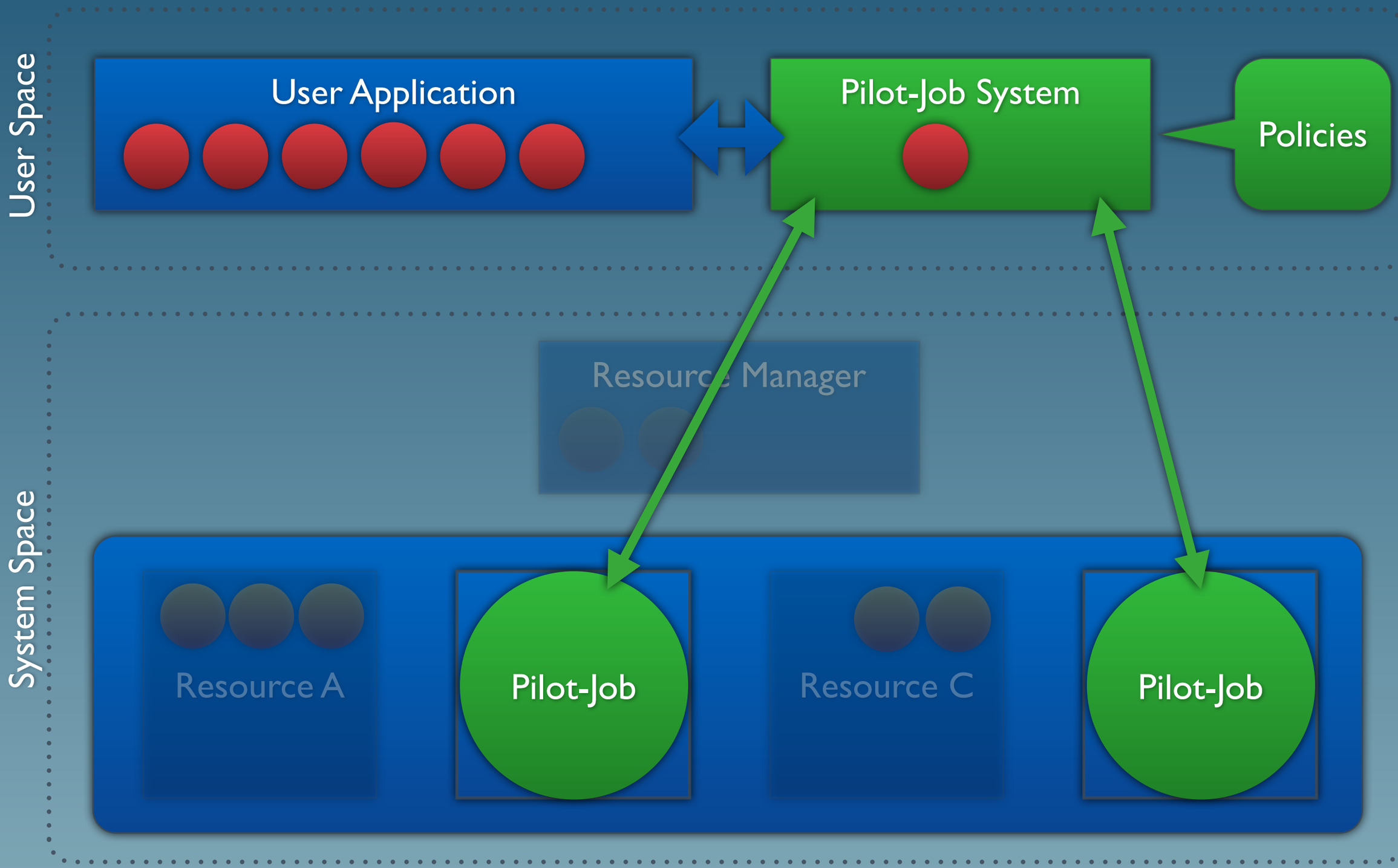
2. Pilot-Job Systems



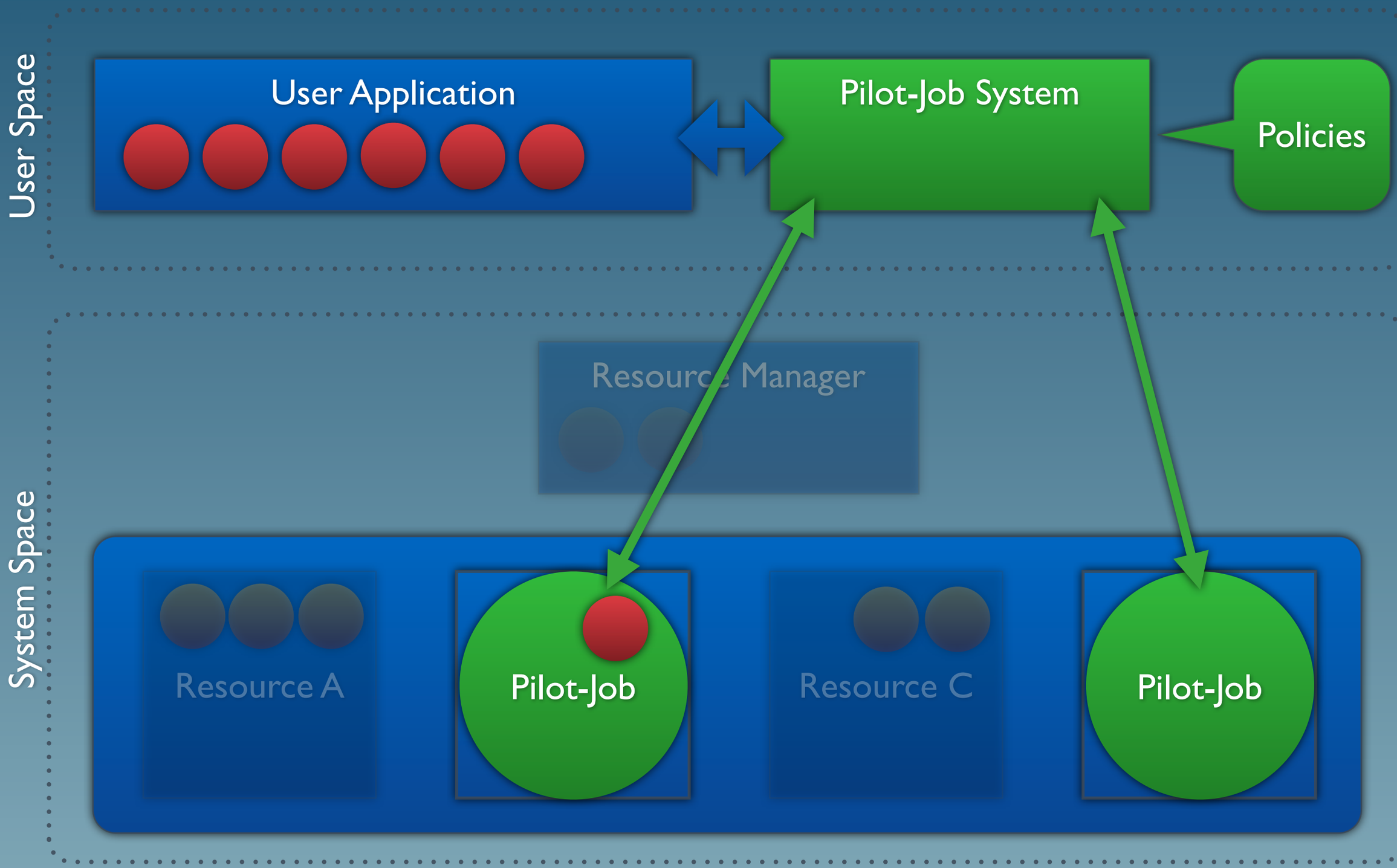
2. Pilot-Job Systems



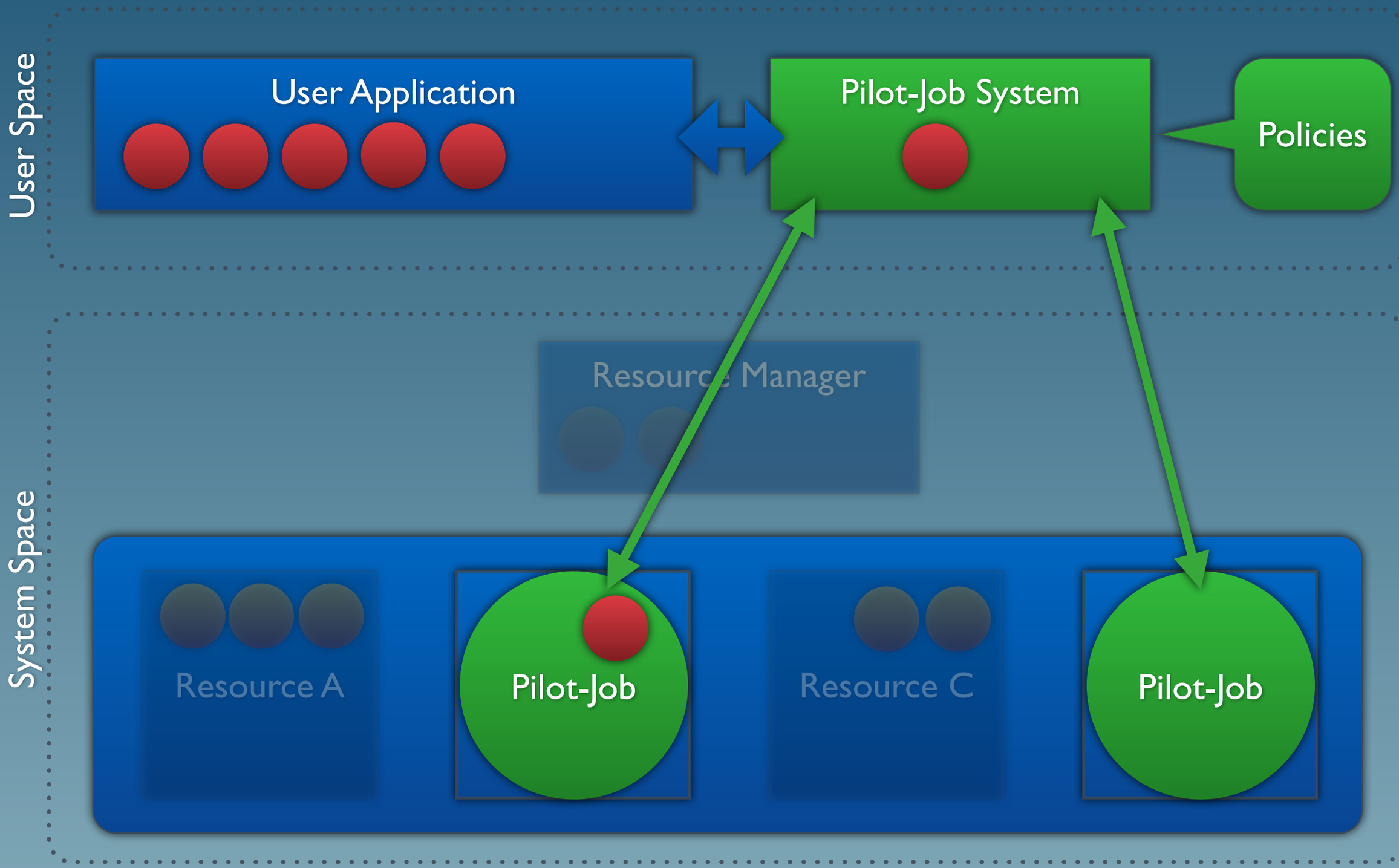
2. Pilot-Job Systems



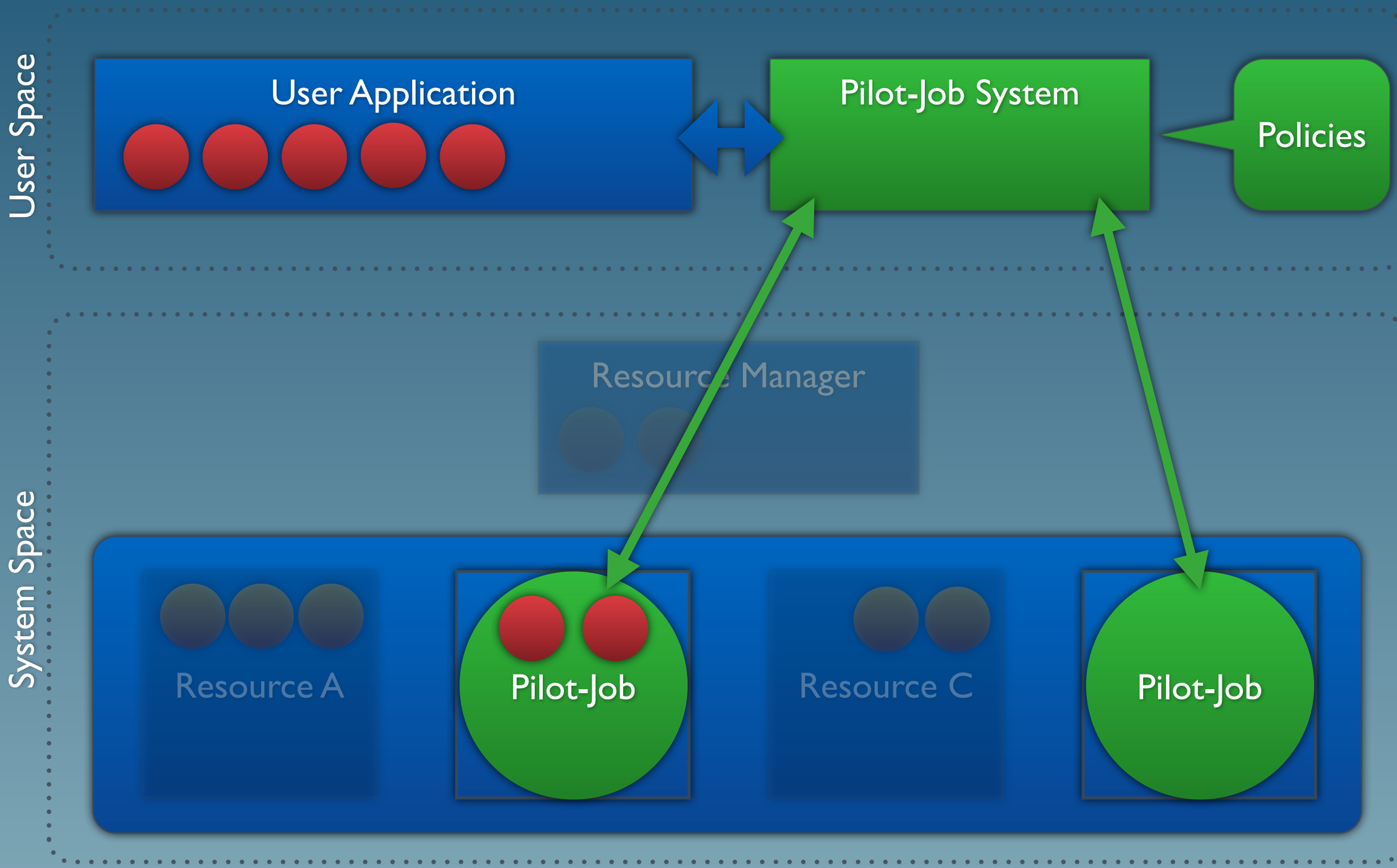
2. Pilot-Job Systems



2. Pilot-Job Systems

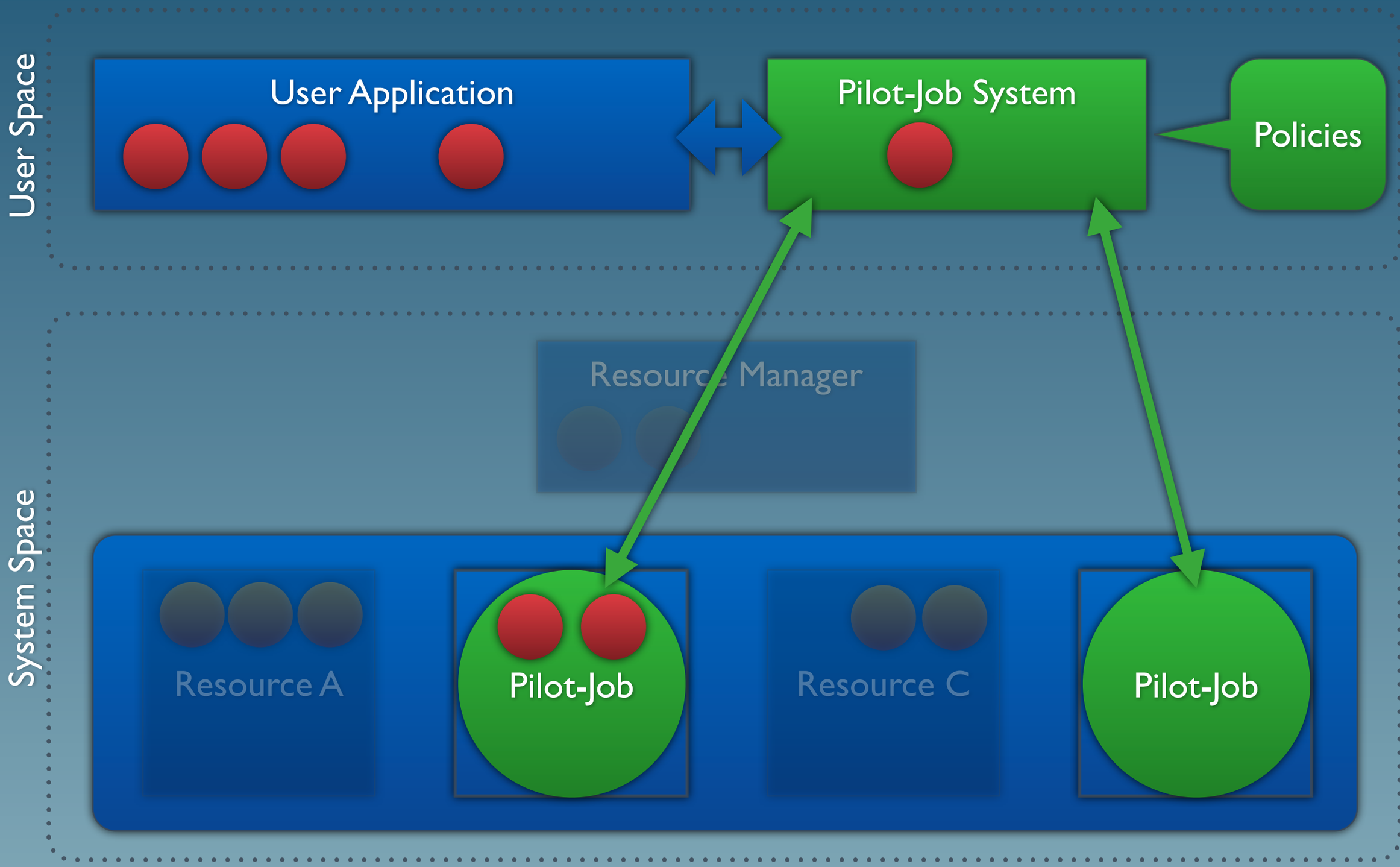


2. Pilot-Job Systems



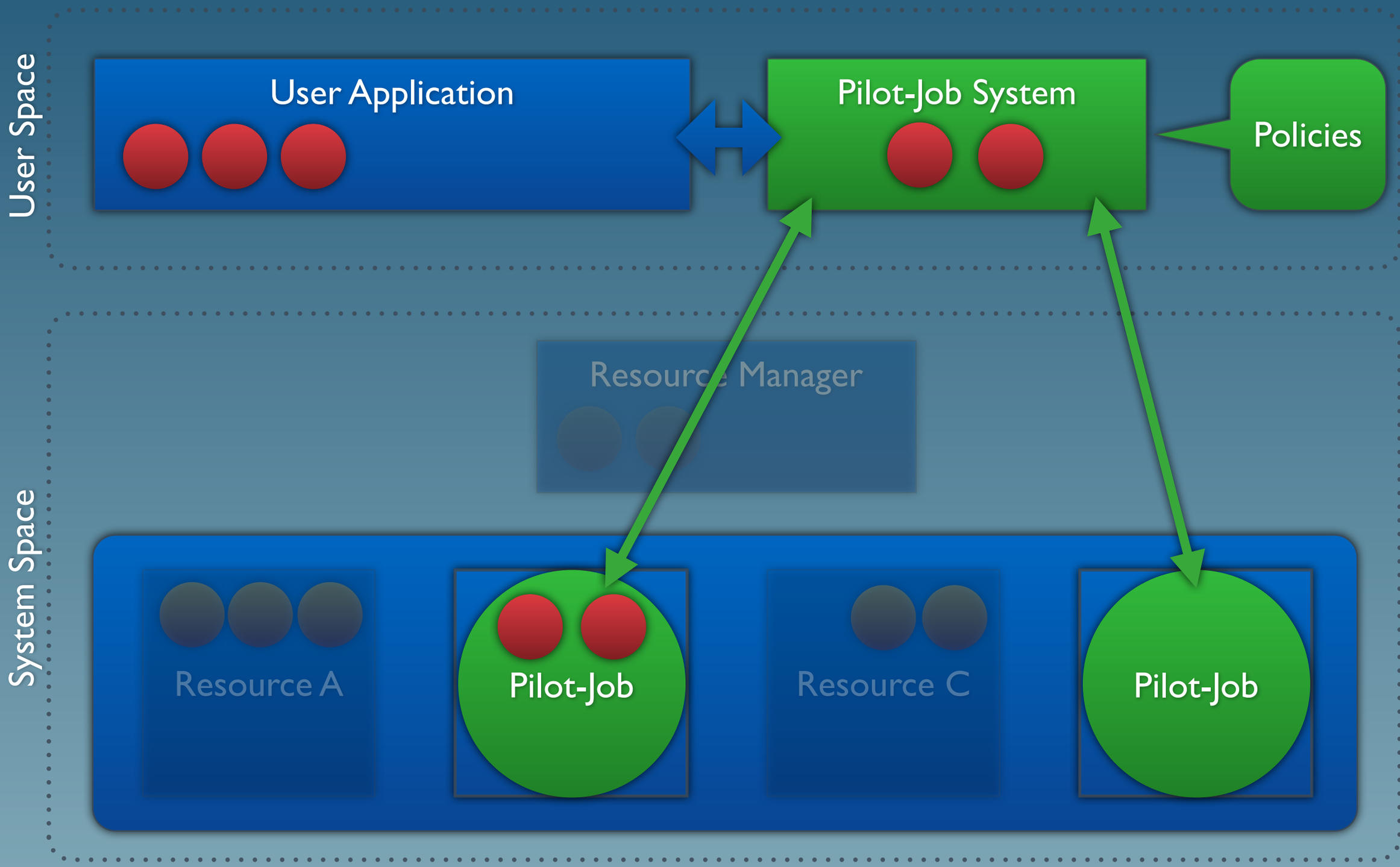
2. Pilot-Job Systems

THE UNIVERSITY of EDINBURGH
informatics
cisa | Centre for Intelligent Systems and their Applications



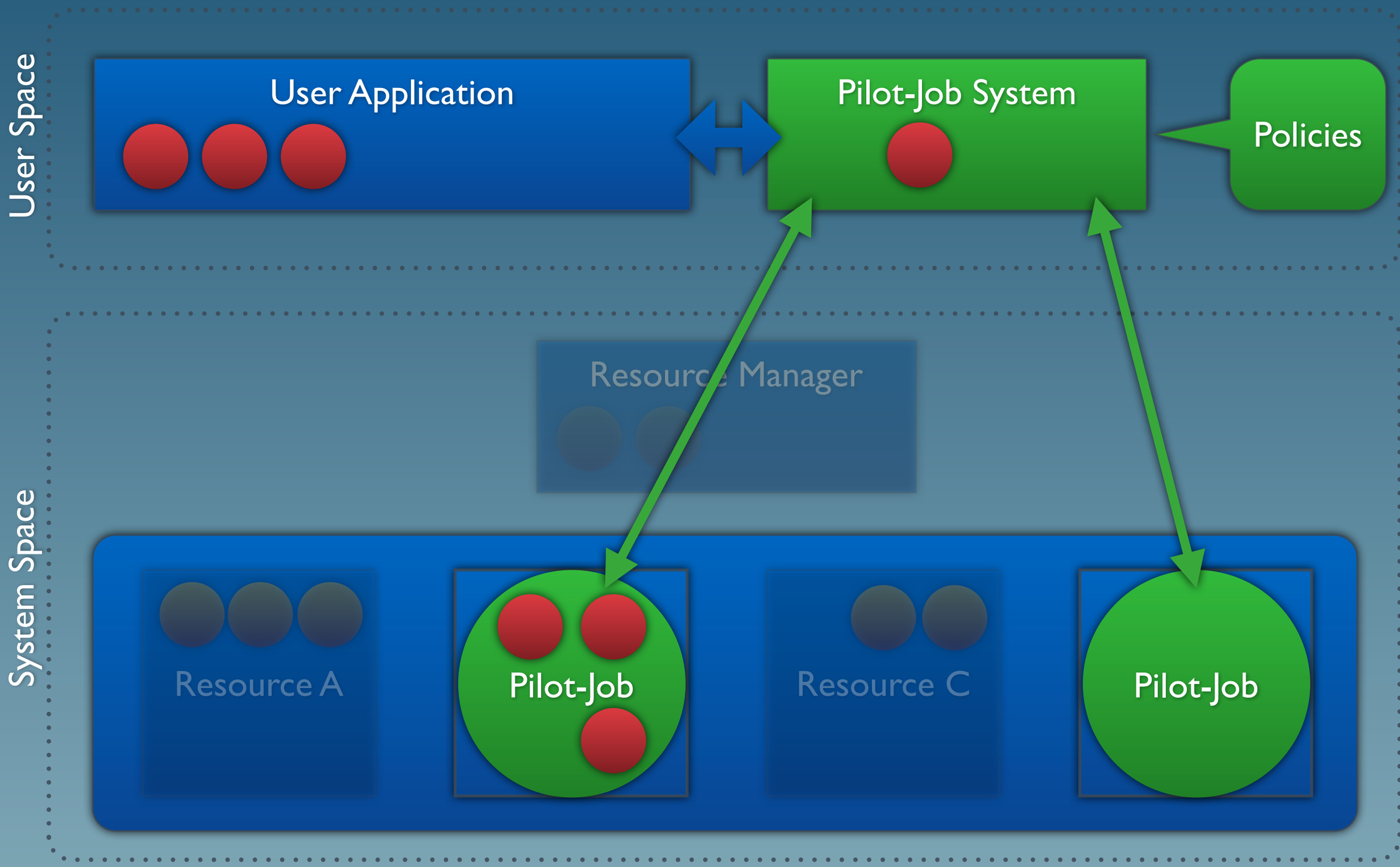
2. Pilot-Job Systems

THE UNIVERSITY of EDINBURGH
informatics
cisa | Centre for Intelligent Systems and their Applications

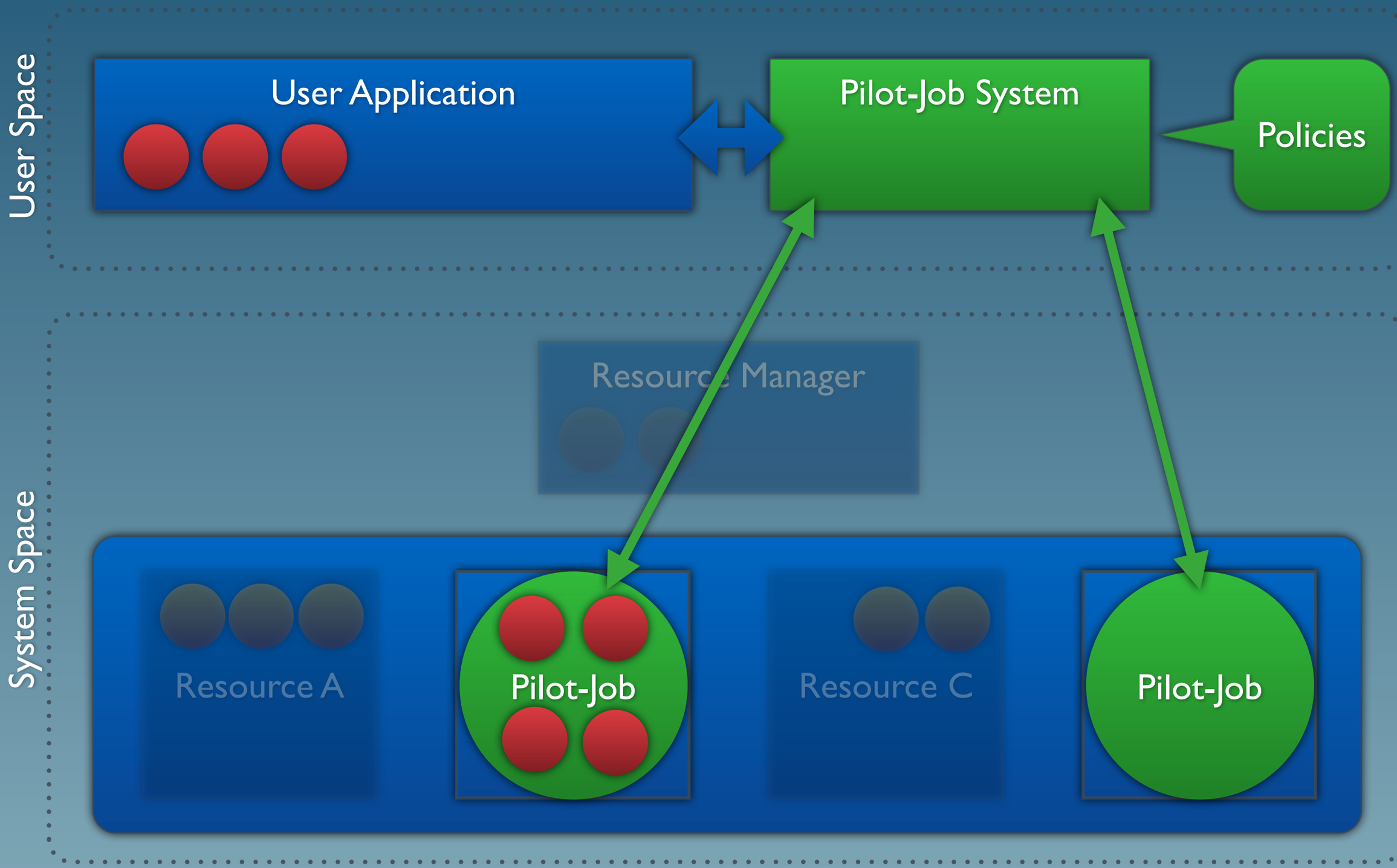


2. Pilot-Job Systems

THE UNIVERSITY of EDINBURGH
informatics
cisa | Centre for Intelligent Systems and their Applications



2. Pilot-Job Systems



3. Pilot-Job Systems

- PJS come in many shapes and forms
  - GridBot, Condor G/Glide-In, Nimrod/G, MyCluster, Falkon, Diane, DIRAC, BigJob, ...
- Useful properties to categorise PJS
  - Communication / Coordination patterns
  - Application / user multiplicity per PJS instance
  - Supported resources / resource managers
  - Scheduling capabilities
- Special case: Pilot-job system becomes part of the system space again (e.g., OSG glideinWMS , EGI)

3. Application Use-Cases

- Example: Queuing time and scheduling / management overhead for large numbers of short-running jobs
  - e.g., CyberShake (seismic hazard model)
    Per run: 840,000 jobs running ~30 seconds each
  - e.g., Montage (space image mosaic workflow)
    Per run: 4,000 jobs running ~60 seconds each

3. Application Use-Cases

- Example: Data locality and affinity
  - e.g., BFAST genome matching: thousands of jobs ~ 10 minutes each, but (depending on reference) 10 to 100 GB input data size
- Example: Scale-'across' applications over multiple distributed resources
  - e.g., cross-institutional collaborations. Different systems, different policies, different interfaces...

4. A Case for Optimsation

- Distributed applications and resource environments are often complex and dynamic. This leaves a lot of room for or even requires optimising task and data placement
- What is the state of the art in application optimization?
  - Manual experiments (test runs) carried out by the user
  - Static optimisation methods based on exp. results
- Conceptual understanding of how to optimise applications automatically exists but is often not applied because of many technical barriers
- Therefore, many applications don't use existing resource efficiently or perform sub-optimally

4. A Case for Optimsation

- Pilot-job systems give applications the practical ability to have better control over task and data scheduling and placement. They are widely used.
- However, introducing multi-level scheduling adds yet another layer of complexity to the application space:
  - Which and what type of resources should be used for resource overlays in a specific application scenario?
  - Just because we *have* more control over scheduling and placement, we still have to find out *what* to do in a specific application scenario / what strategy to use
  - How do we deal with applications that exhibit *dynamic* properties (e.g., data)? Adaptive optimisation?

5. Application Optimsation on PJS-Level

THE UNIVERSITY of EDINBURGH
informatics
cisa | Centre for Intelligent Systems and their Applications

- Pilot-job systems provide a context to evaluate, explore and apply new and existing optimisation strategies in a broader application and resource environment
  - Provisioning and scheduling is decoupled. Resources are uniformly abstracted
  - PJS operate on workload- / task-level, i.e., on PJS level application workload (task+data) is uniformly abstracted
- Feasibility hasn't been systematically explored before in a PJS context
  - PJS are still mostly seen as a HTC on HPC solution
  - Few point solutions show PJS-level optimisation

5. Application Optimsation on PJS-Level

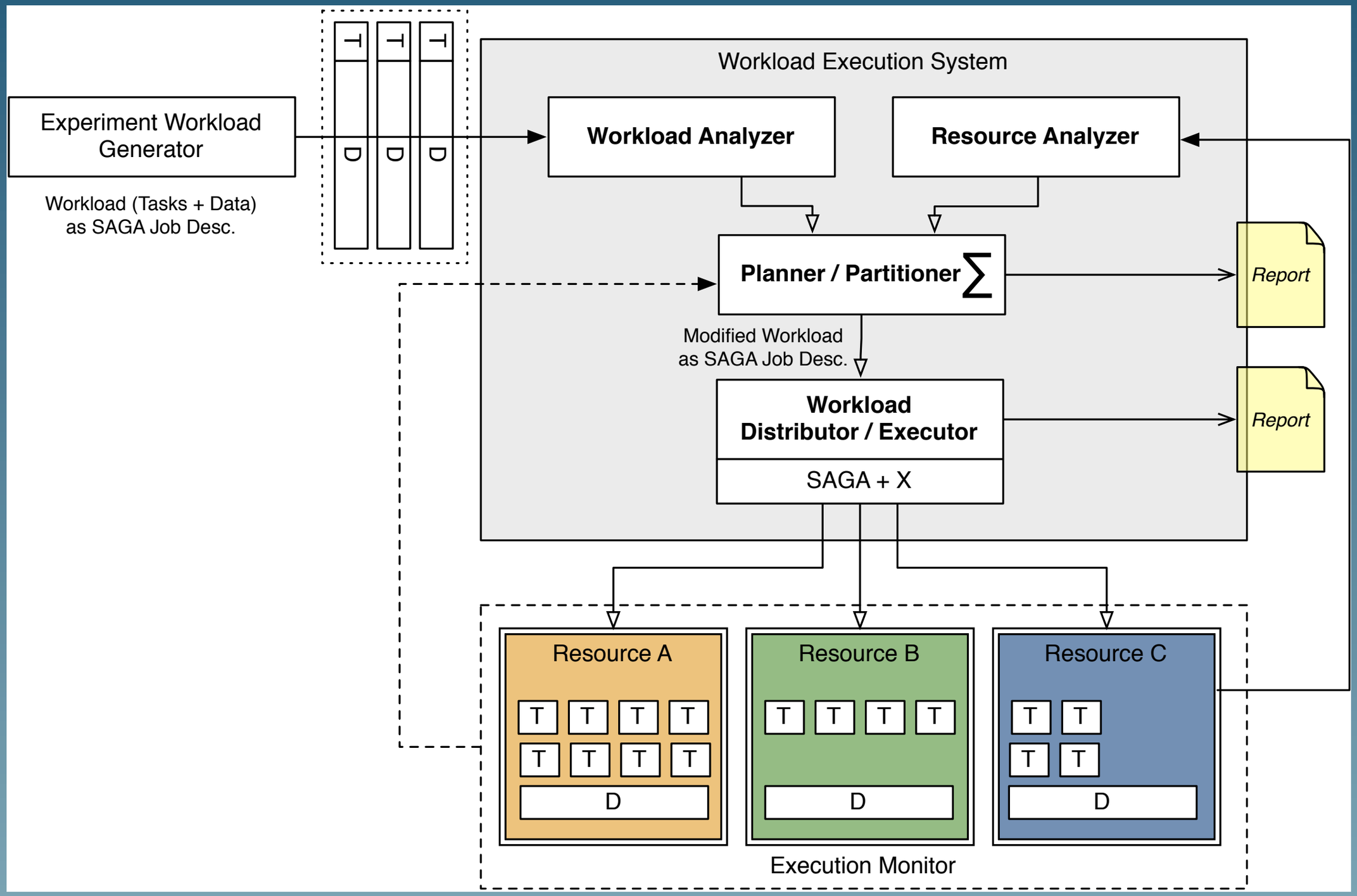
THE UNIVERSITY of EDINBURGH
informatics
cisa | Centre for Intelligent Systems and their Applications

- Optimisation on PJS level requires to explore and understand the following:
  - Is enough information available on PJS level to support different optimisation techniques, and if so:
  - What are the relevant application metrics for task and data placement decisions?
    - How can we capture / extract those metrics on workload level?
  - What are the relevant resource metrics?
    - How can we measure and interpret them?
  - How do we express optimisation goals and strategies

6. Integration with a Pilot-Job System

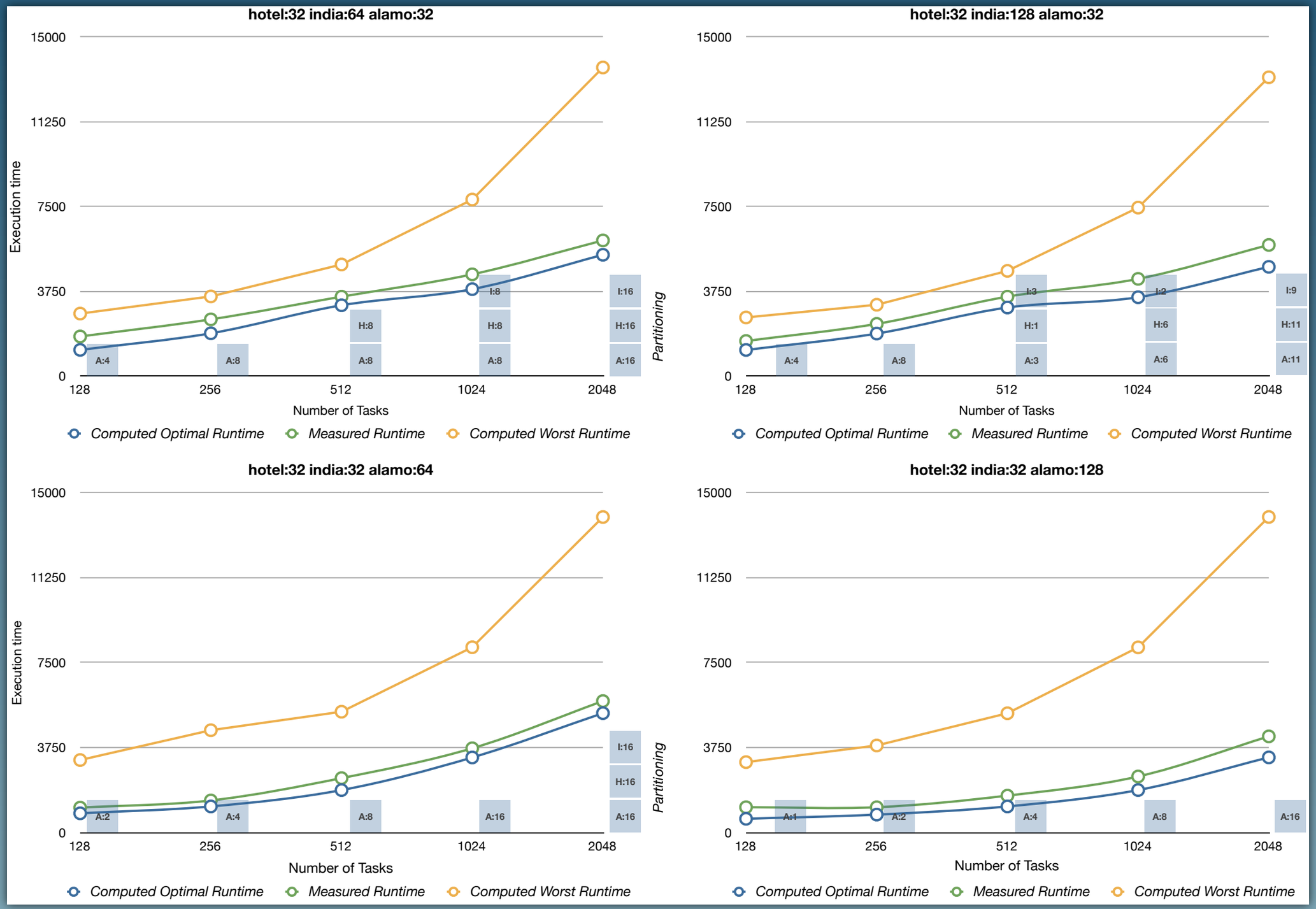
- BigJob: A SAGA-based pilot-job system
  - Uses SAGA as resource access layer for the resource provisioning component (can talk to PBS, Globus, Condor, Amazon EC2, ...)
  - Uses SAGA/JSDL based workload description
  - Light-weight agents written in Python are easily extensible to integrate capturing of resource metrics
- Goal: Try to integrate an optimisation mechanism (e.g., throughput optimisation) into the BigJob framework
  - Experimentally explore feasibility
  -

6. Integration with a Pilot-Job System



6. Integration with a Pilot-Job System

THE UNIVERSITY of EDINBURGH
informatics
cisa | Centre for Intelligent Systems and their Applications



6. Integration with a Pilot-Job System

THE UNIVERSITY of EDINBURGH
informatics
cisa | Centre for Intelligent Systems and their Applications

- Next steps
  - Evaluate with multiple applications
  - Implement alternative optimisation techniques
  - Explore ways to abstractly define optimisation techniques

Questions