Performance Isolation and Fairness for Multi-Tenant Cloud Storage

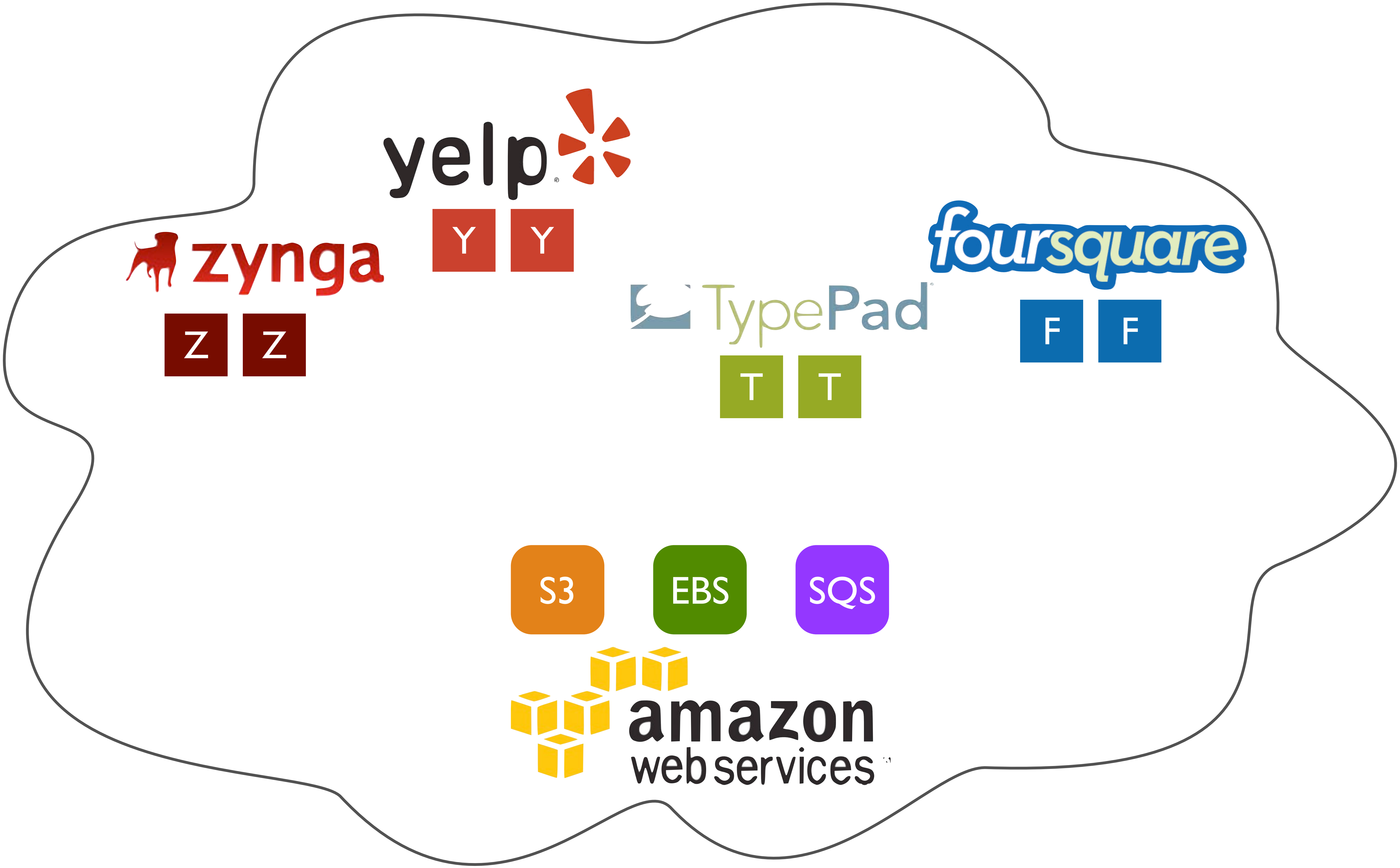
PISCES

David Shue*, Michael Freedman*, and Anees Shaikh✦

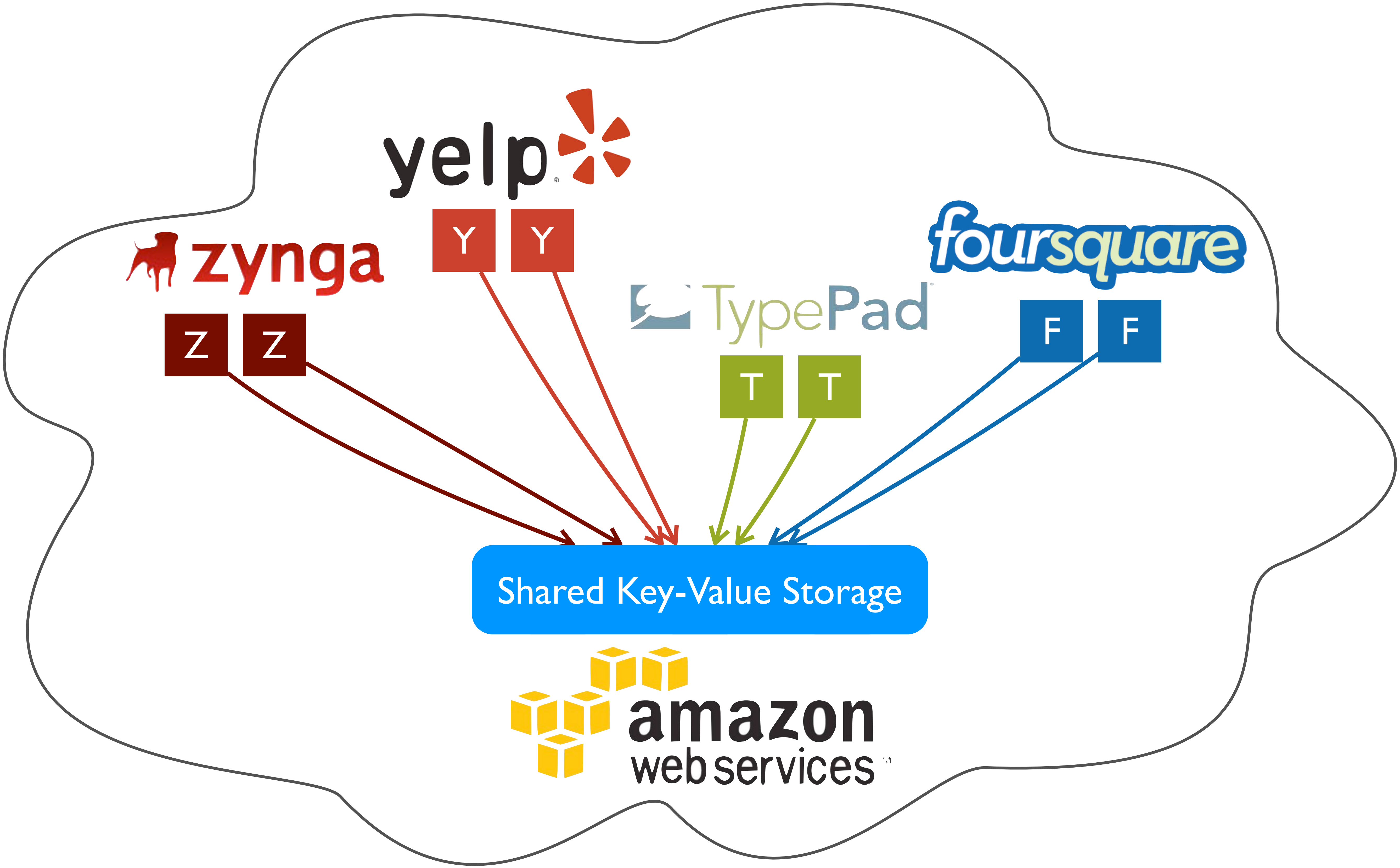
*Princeton ✦IBM Research

Setting: Shared Storage in the Cloud

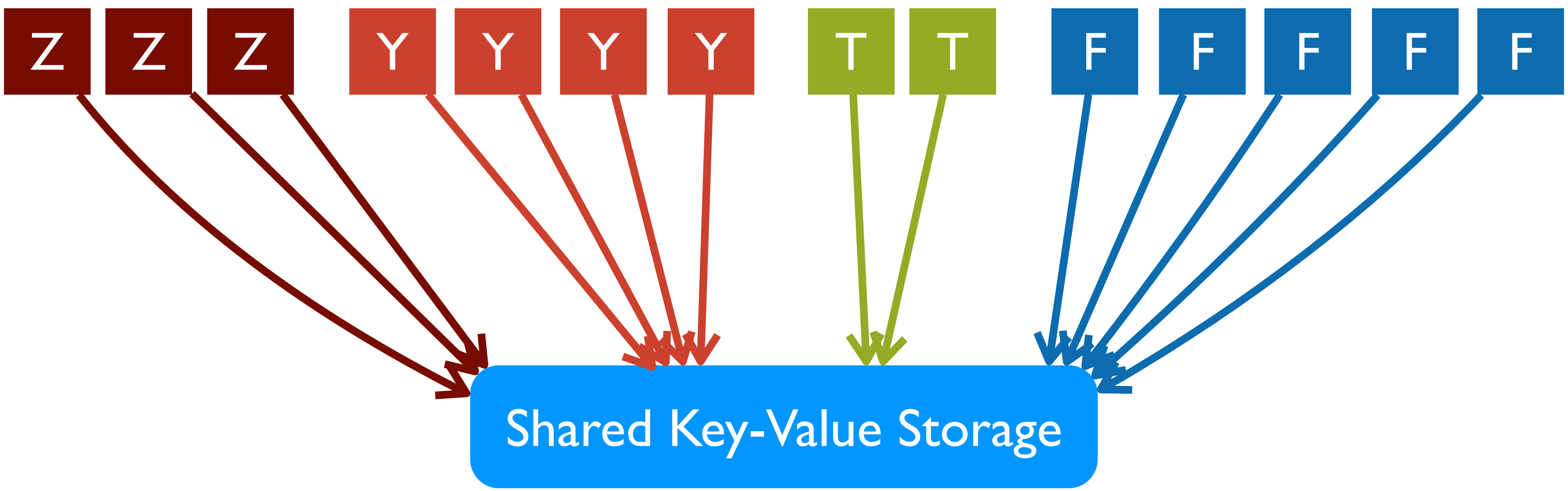
Setting: Shared Storage in the Cloud



Setting: Shared Storage in the Cloud

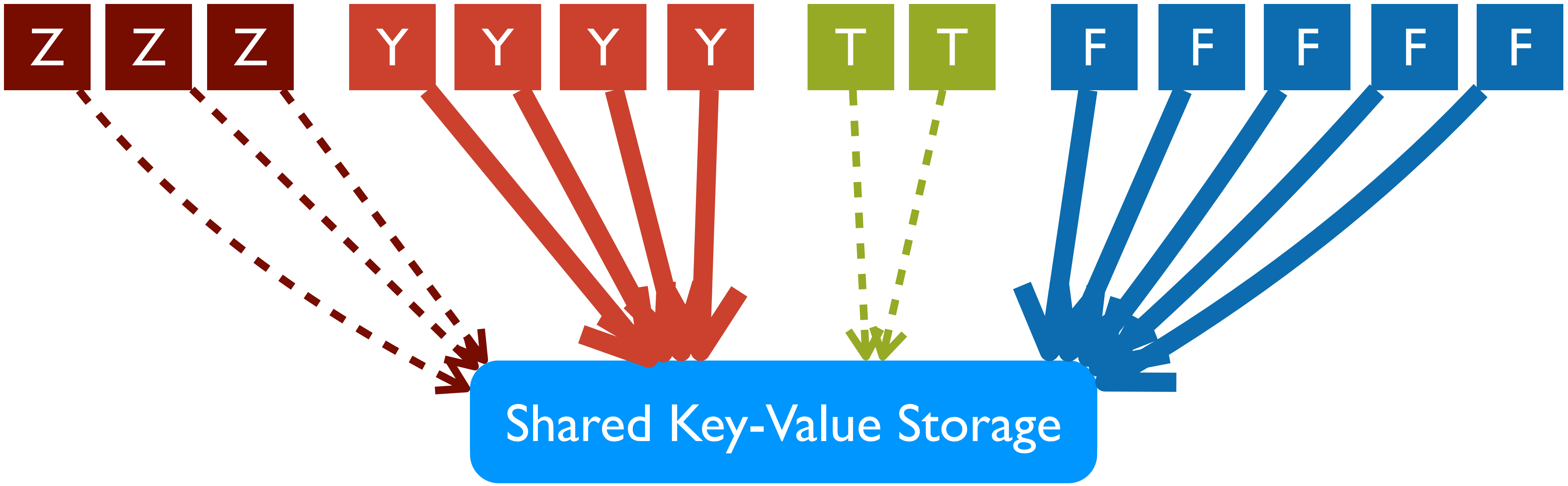


Predictable Performance is Hard



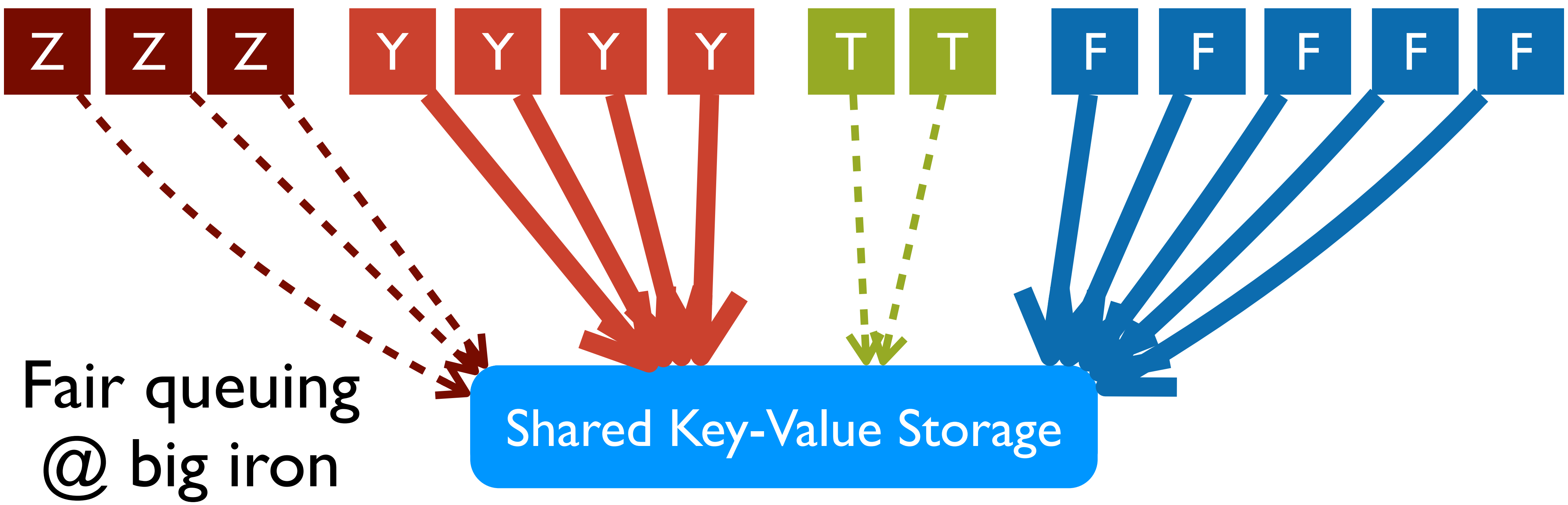
Multiple co-located tenants ⇒ resource contention

Predictable Performance is Hard



Multiple co-located tenants ⇒ resource contention

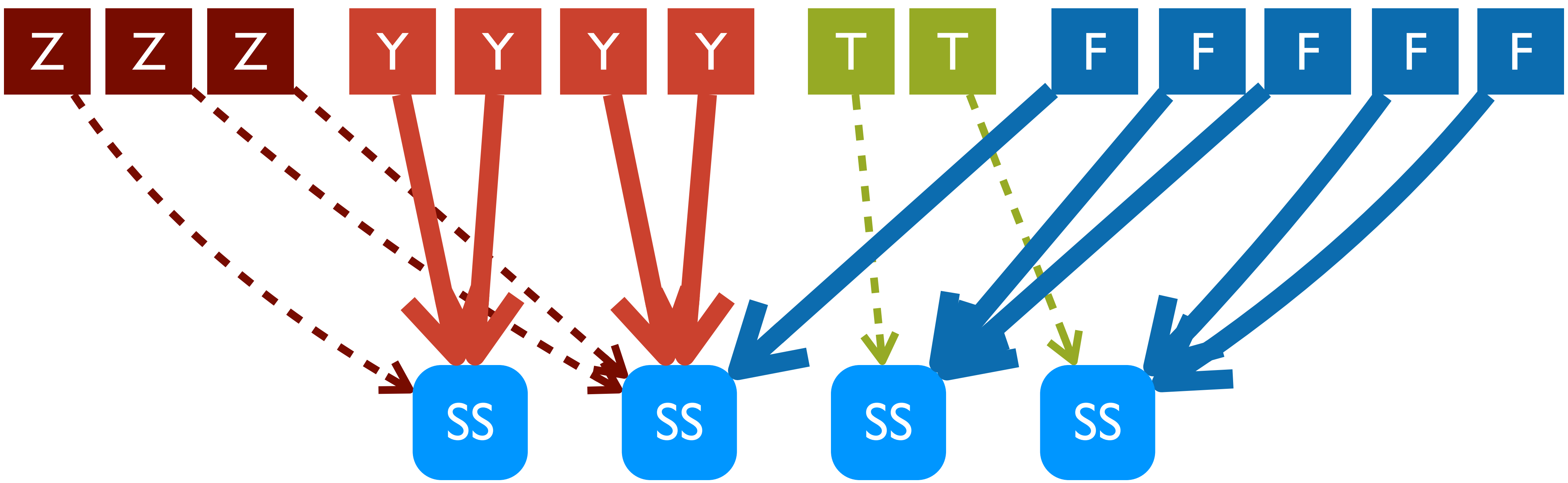
Predictable Performance is Hard



Fair queuing
@ big iron

Multiple co-located tenants ⇒ resource contention

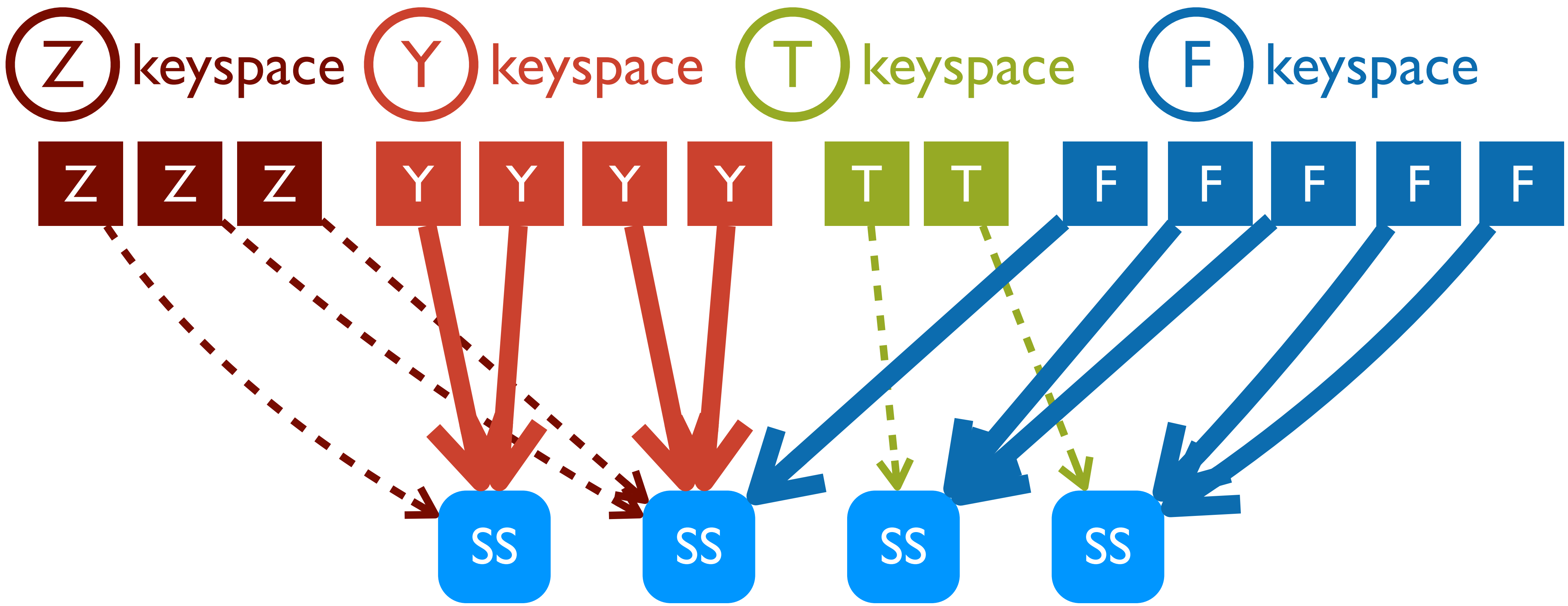
Predictable Performance is Hard



Multiple co-located tenants ⇒ resource contention

Distributed system ⇒ distributed resource allocation

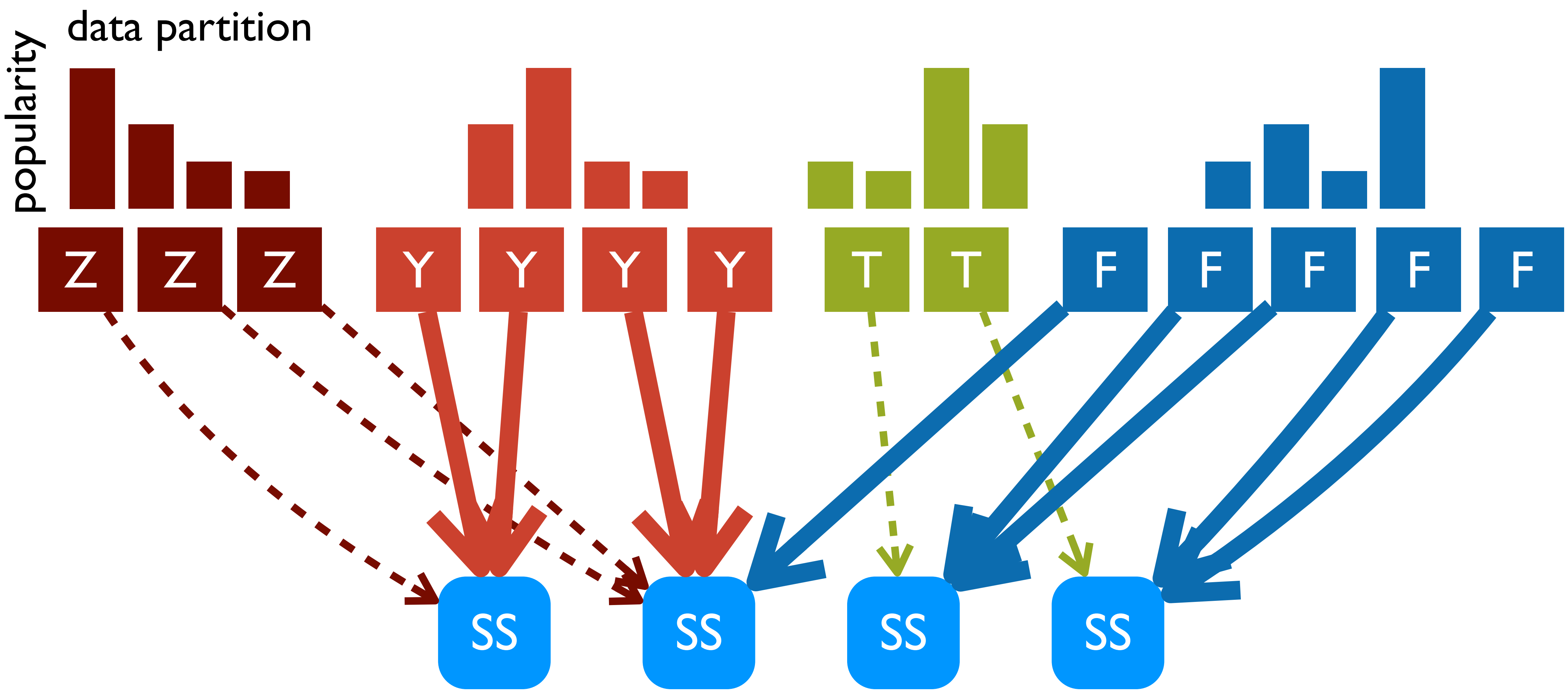
Predictable Performance is Hard



Multiple co-located tenants ⇒ resource contention

Distributed system ⇒ distributed resource allocation

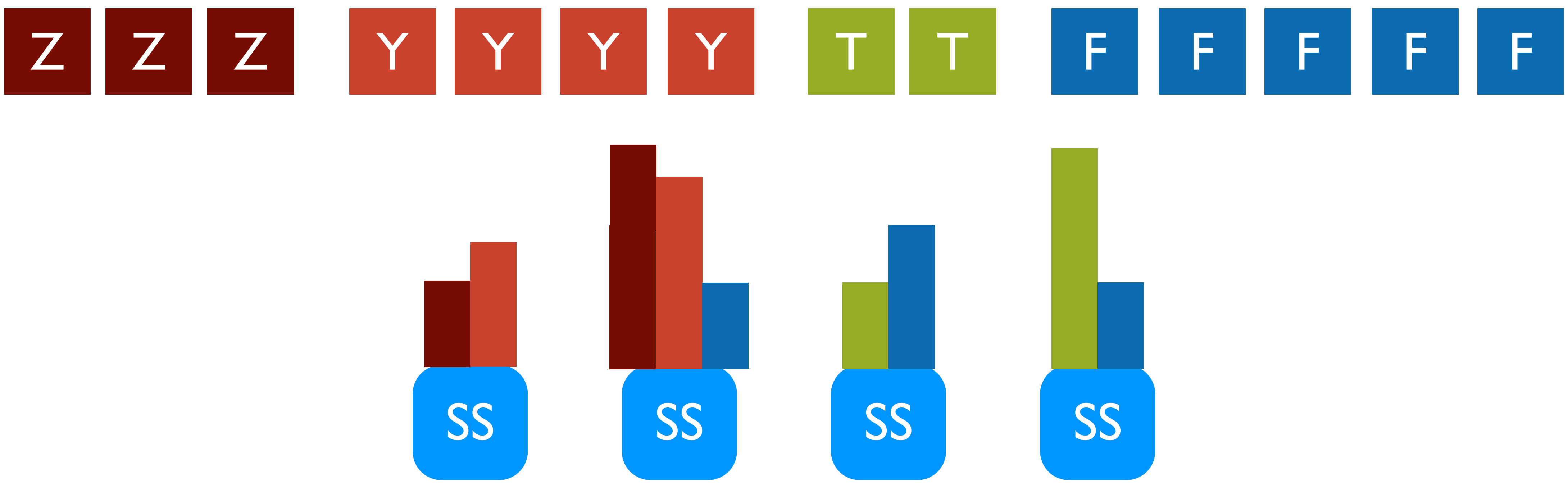
Predictable Performance is Hard



Multiple co-located tenants ⇒ resource contention

Distributed system ⇒ distributed resource allocation

Predictable Performance is Hard

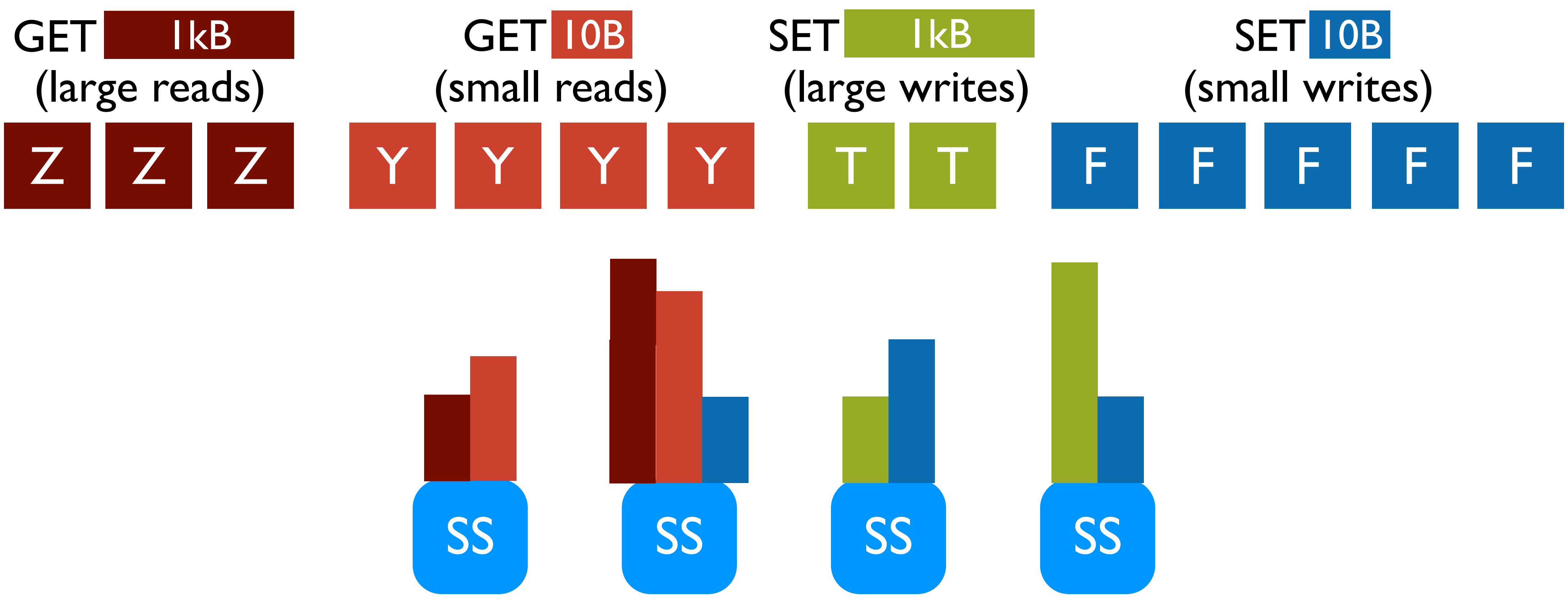


Multiple co-located tenants ⇒ resource contention

Distributed system ⇒ distributed resource allocation

Skewed object popularity ⇒ variable per-node demand

Predictable Performance is Hard



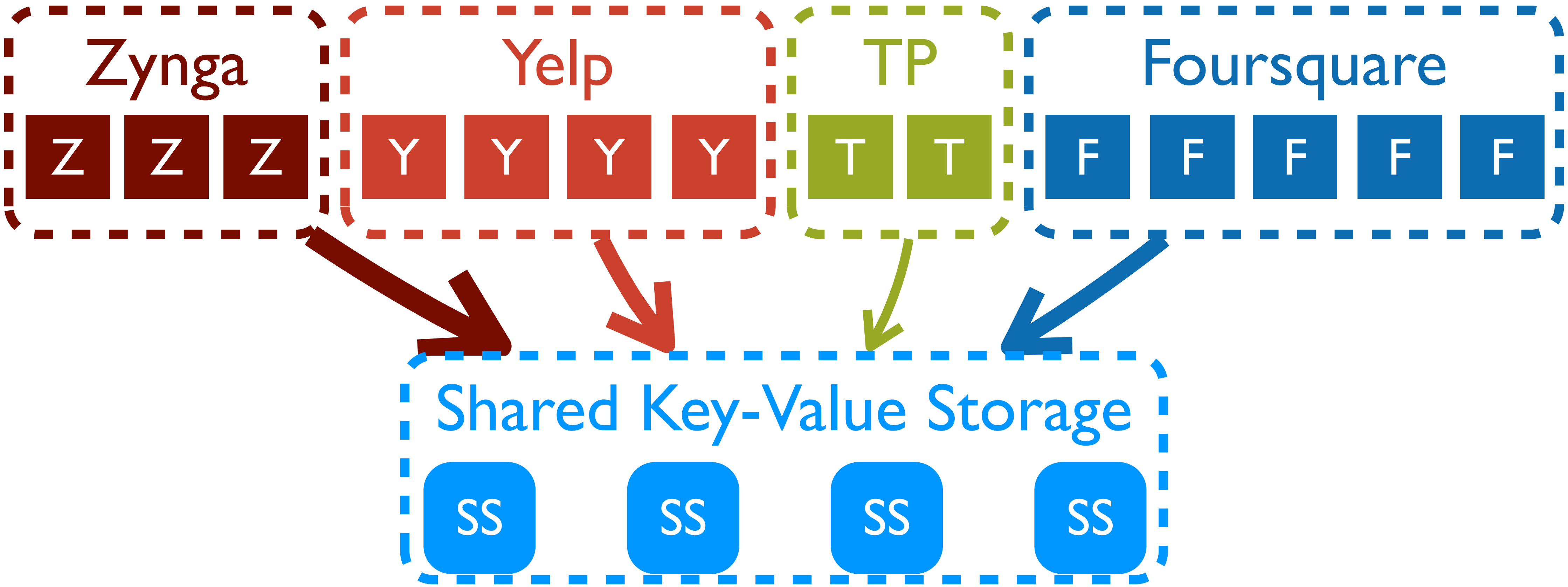
Multiple co-located tenants ⇒ resource contention

Distributed system ⇒ distributed resource allocation

Skewed object popularity ⇒ variable per-node demand

Disparate workloads ⇒ different bottleneck resources

Tenants Want System-wide Resource Guarantees



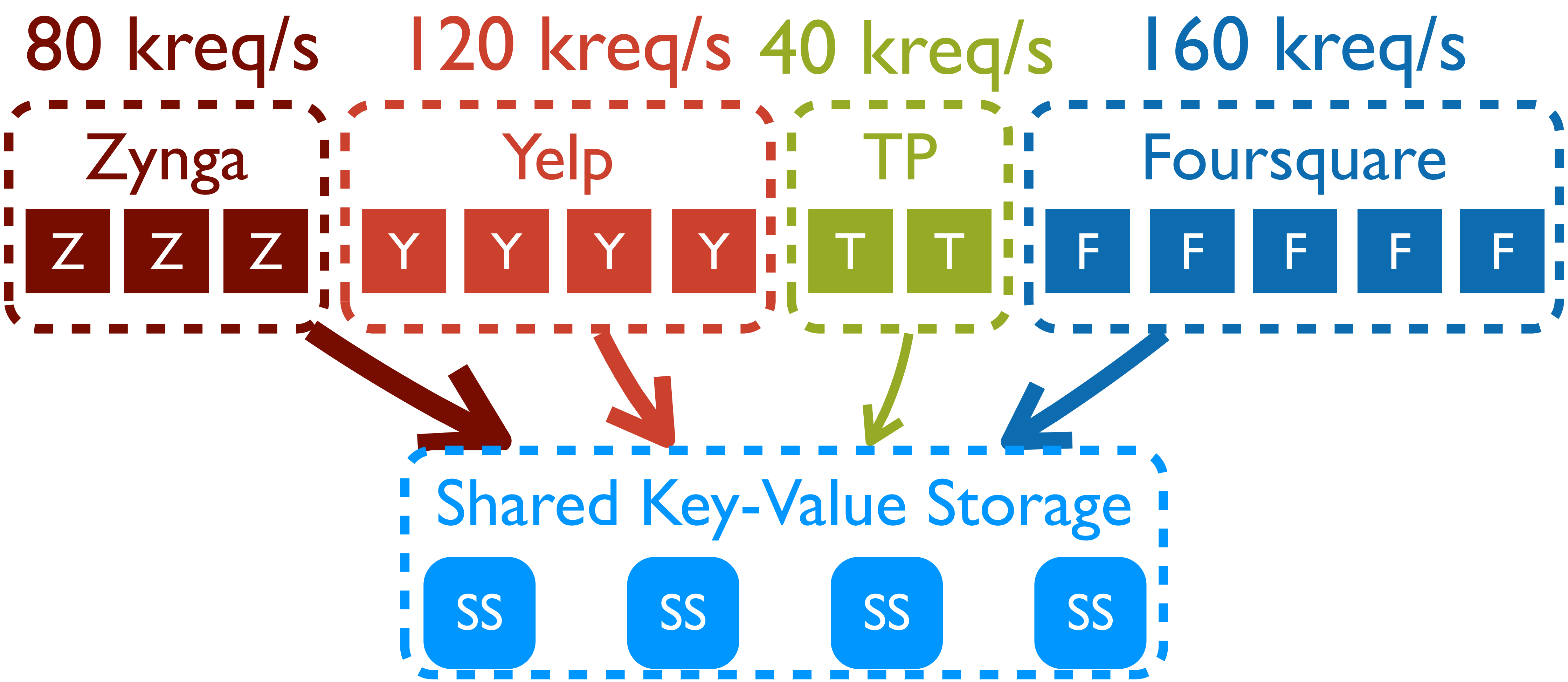
Multiple co-located tenants ⇒ resource contention

Distributed system ⇒ distributed resource allocation

Skewed object popularity ⇒ variable per-node demand

Disparate workloads ⇒ different bottleneck resources

Tenants Want System-wide Resource Guarantees



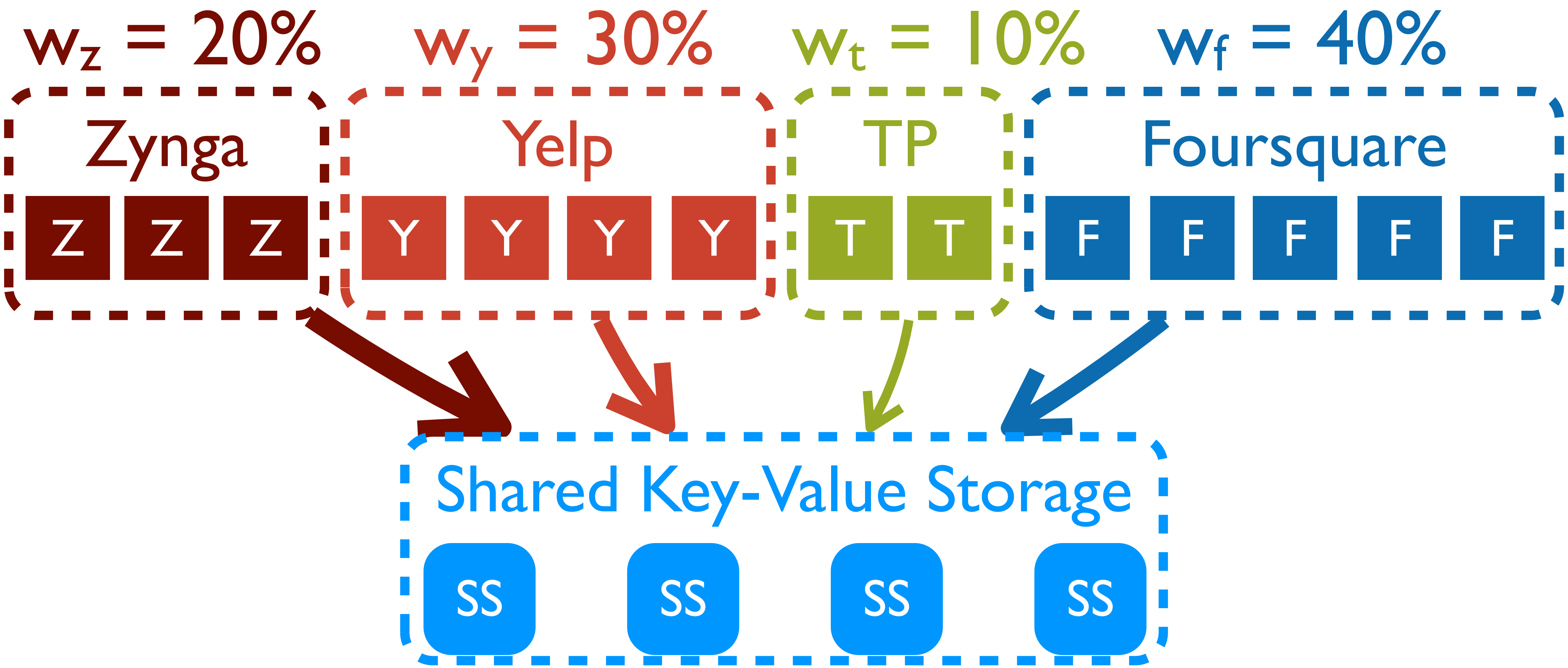
Multiple co-located tenants ⇒ resource contention

Distributed system ⇒ distributed resource allocation

Skewed object popularity ⇒ variable per-node demand

Disparate workloads ⇒ different bottleneck resources

Pisces Provides Weighted Fair-shares



Multiple co-located tenants ⇒ resource contention

Distributed system ⇒ distributed resource allocation

Skewed object popularity ⇒ variable per-node demand

Disparate workloads ⇒ different bottleneck resources

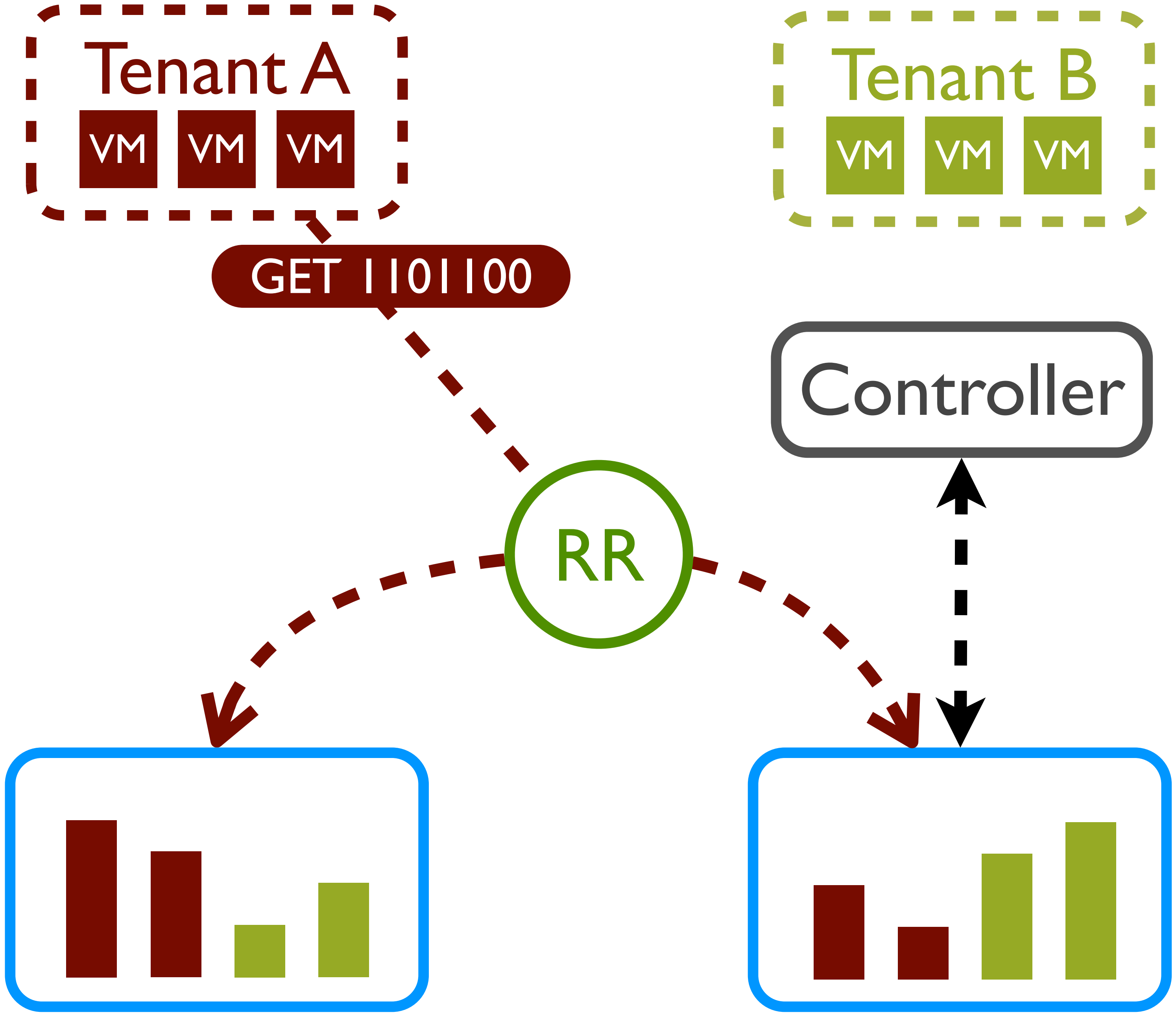
Pisces: Predictable Shared Cloud Storage

- Pisces
  - Per-tenant max-min fair shares of system-wide resources
    ~ min guarantees, high utilization
  - Arbitrary object popularity
  - Different resource bottlenecks

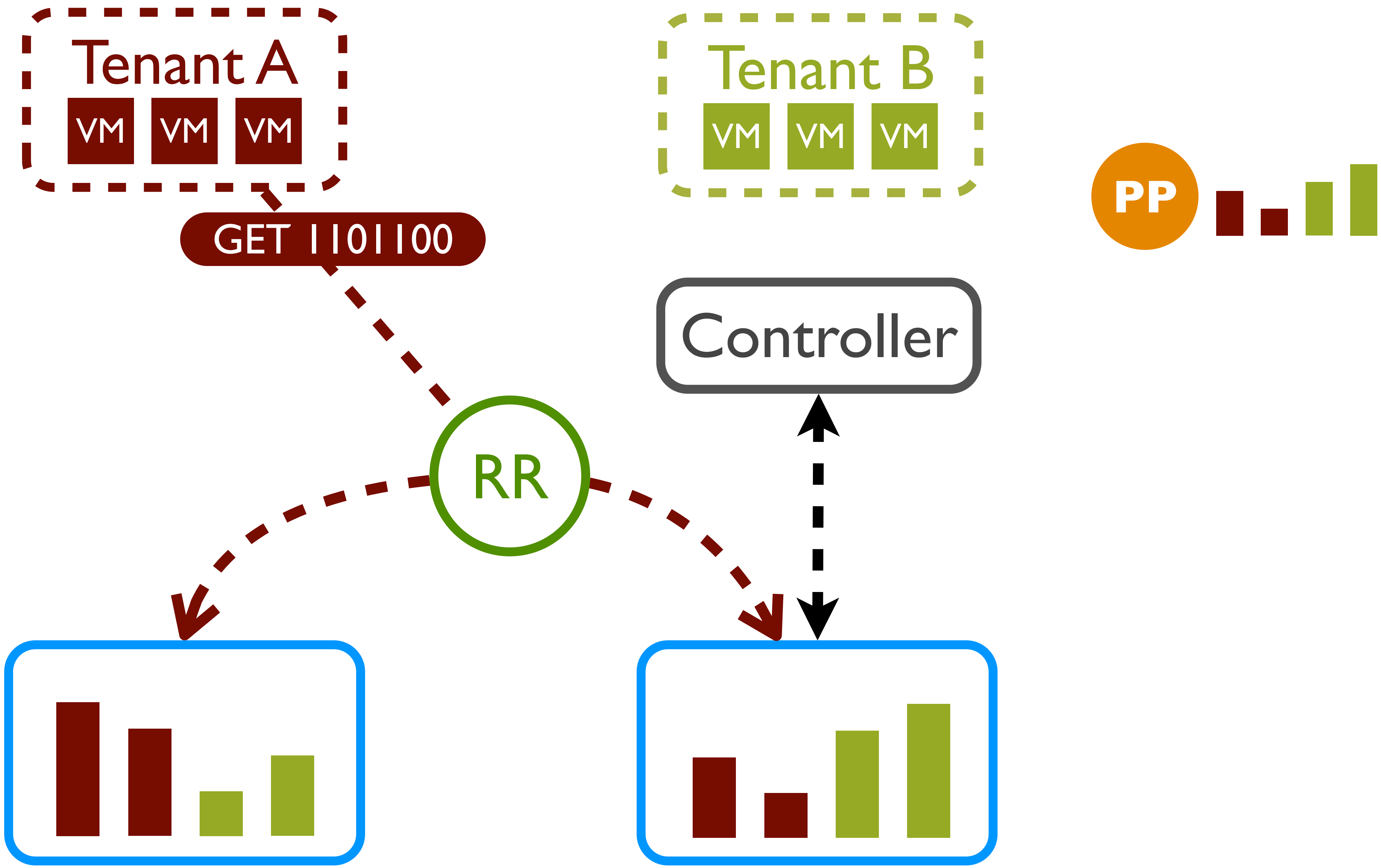
Pisces: Predictable Shared Cloud Storage

- Pisces
  - Per-tenant max-min fair shares of system-wide resources
    ~ min guarantees, high utilization
  - Arbitrary object popularity
  - Different resource bottlenecks
- Amazon DynamoDB
  - Per-tenant provisioned rates
    ~ rate limited, non-work conserving
  - Uniform object popularity
  - Single resource (1kB requests)

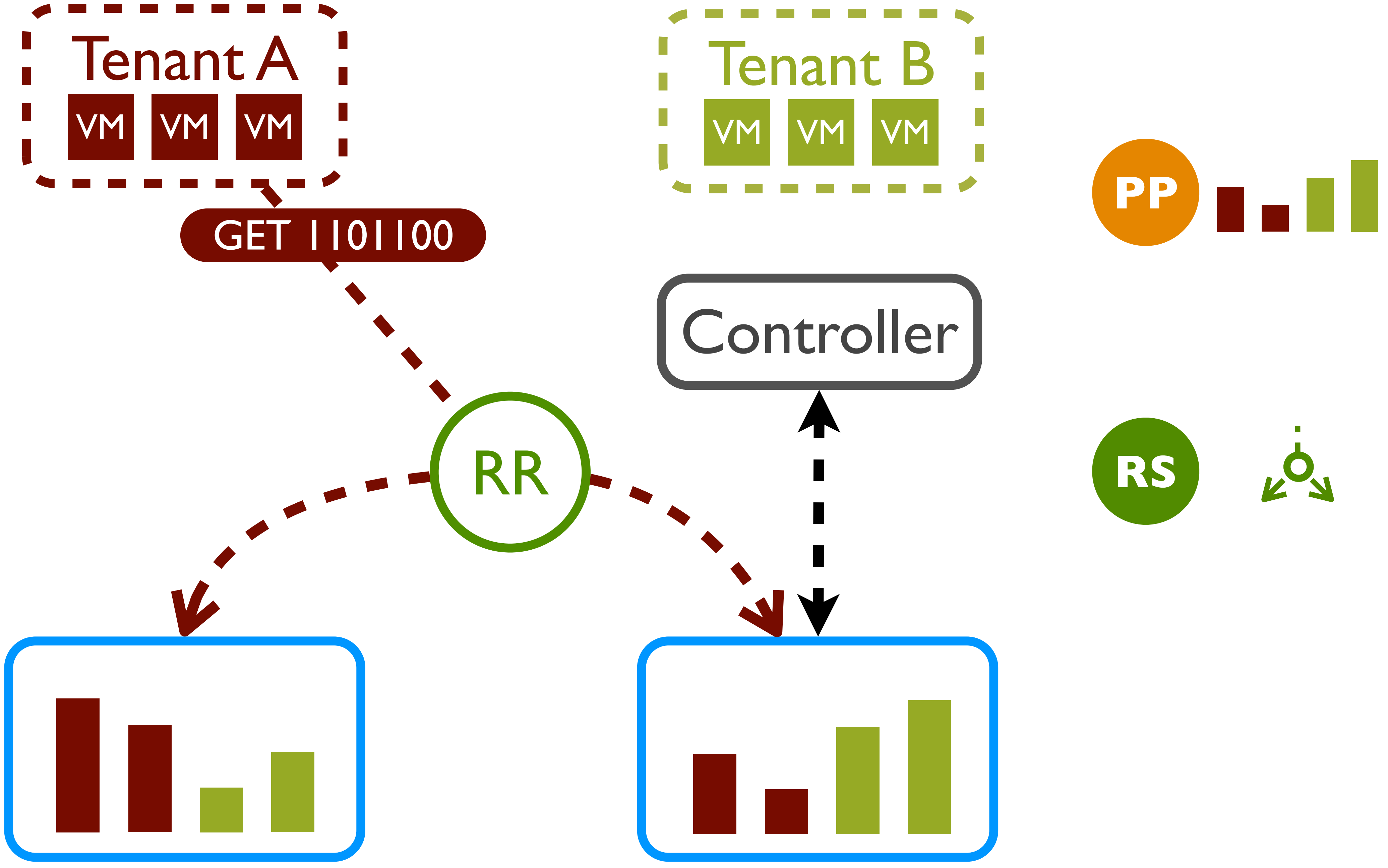
Predictable Multi-Tenant Key-Value Storage



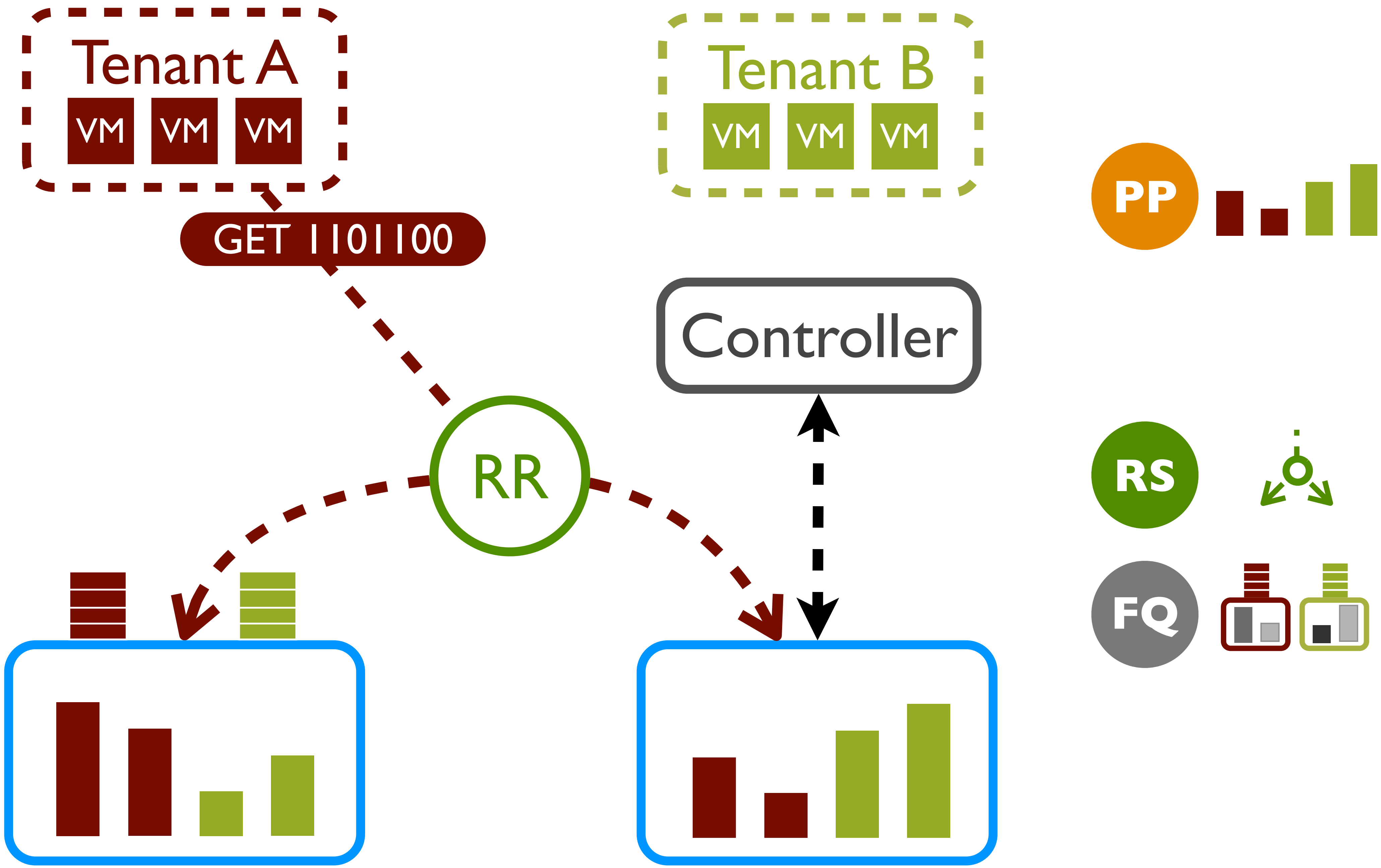
Predictable Multi-Tenant Key-Value Storage



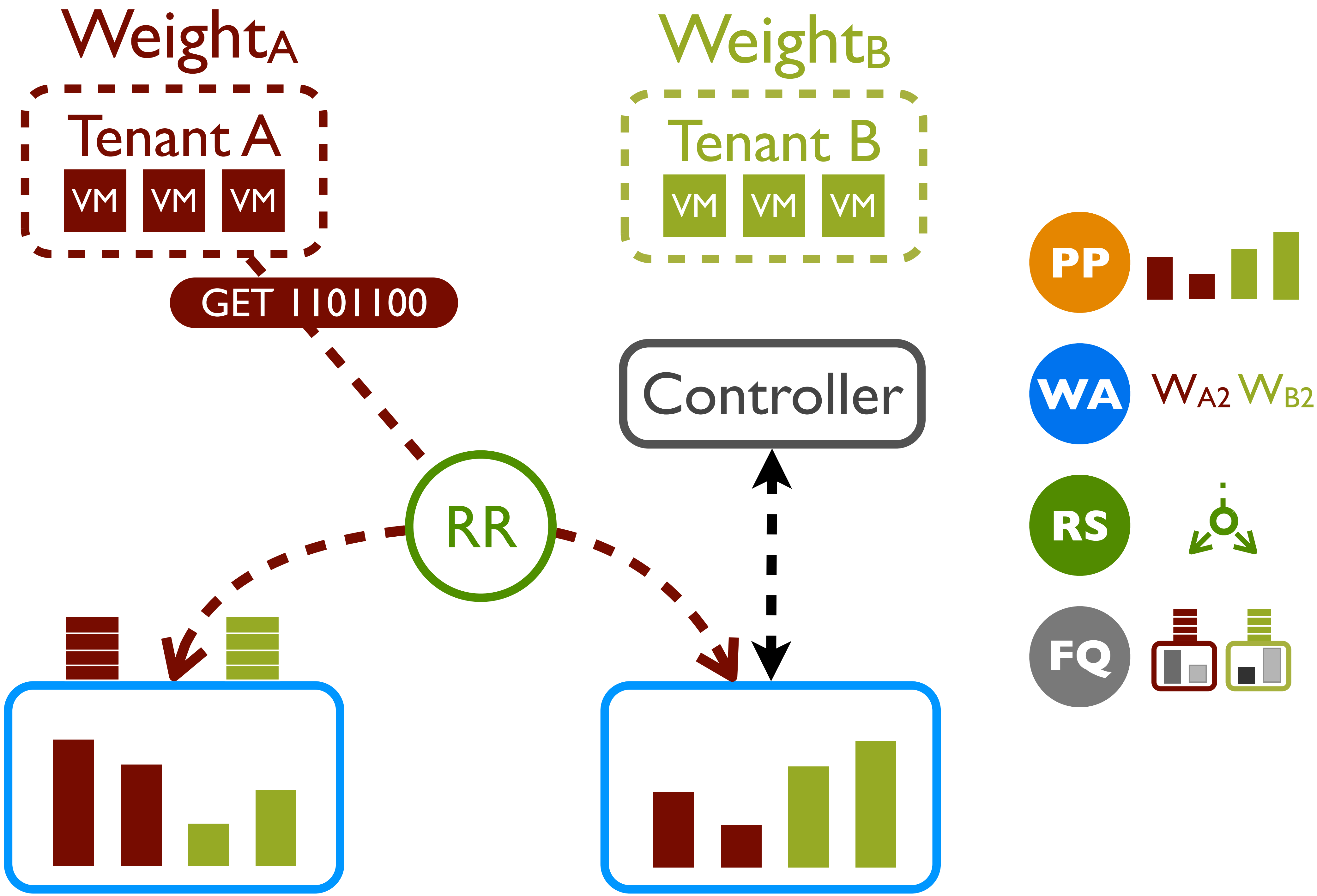
Predictable Multi-Tenant Key-Value Storage



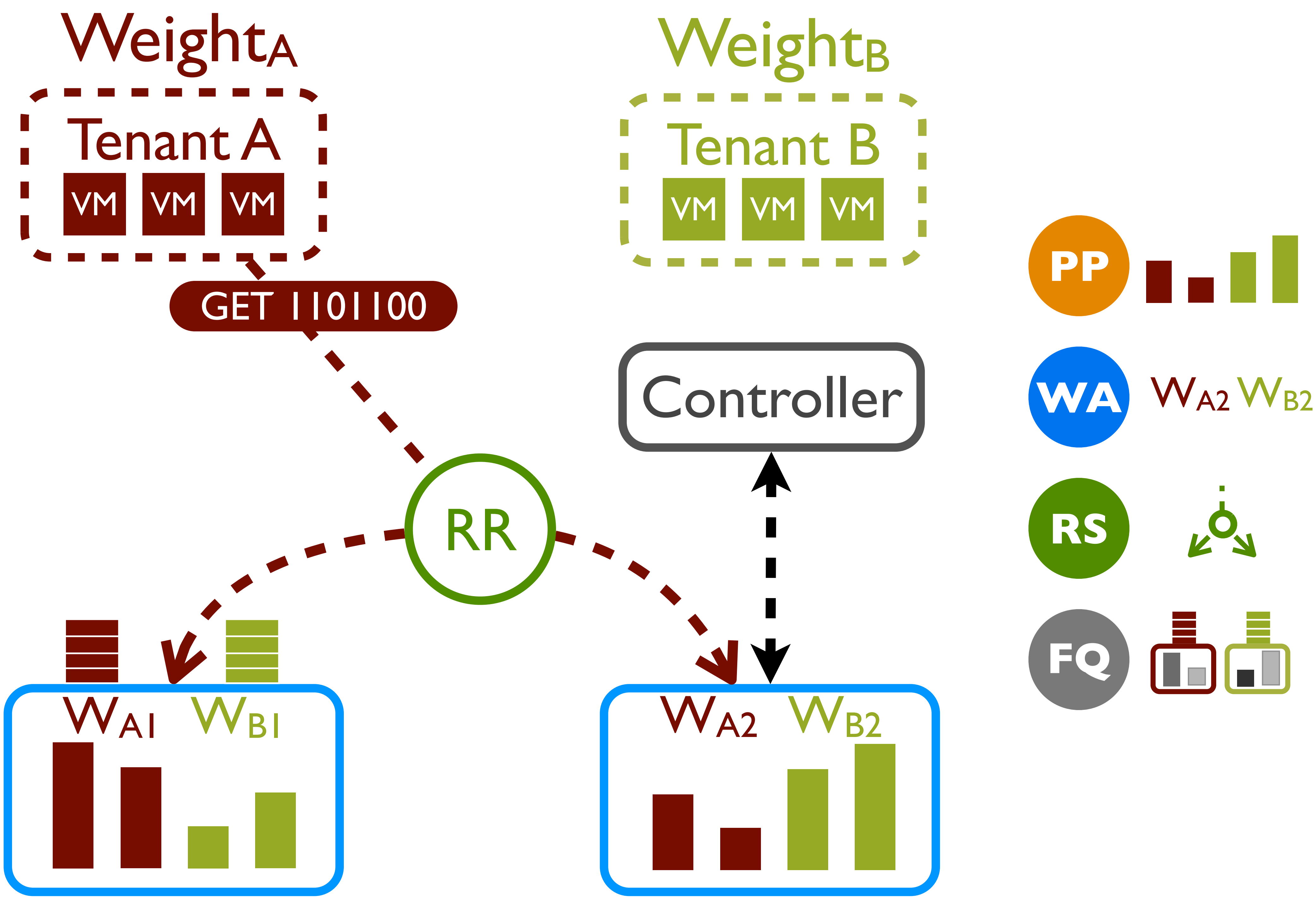
Predictable Multi-Tenant Key-Value Storage



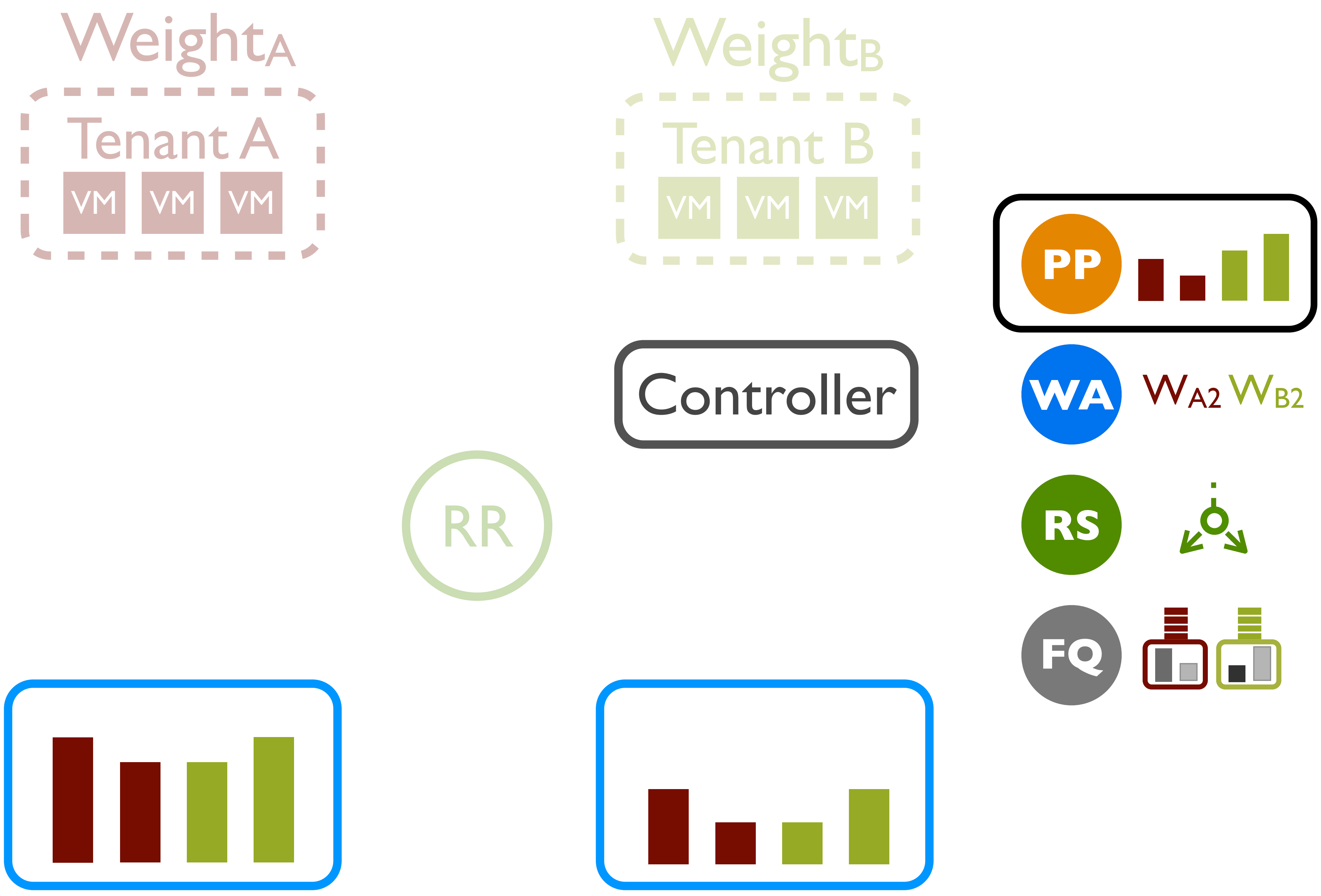
Predictable Multi-Tenant Key-Value Storage



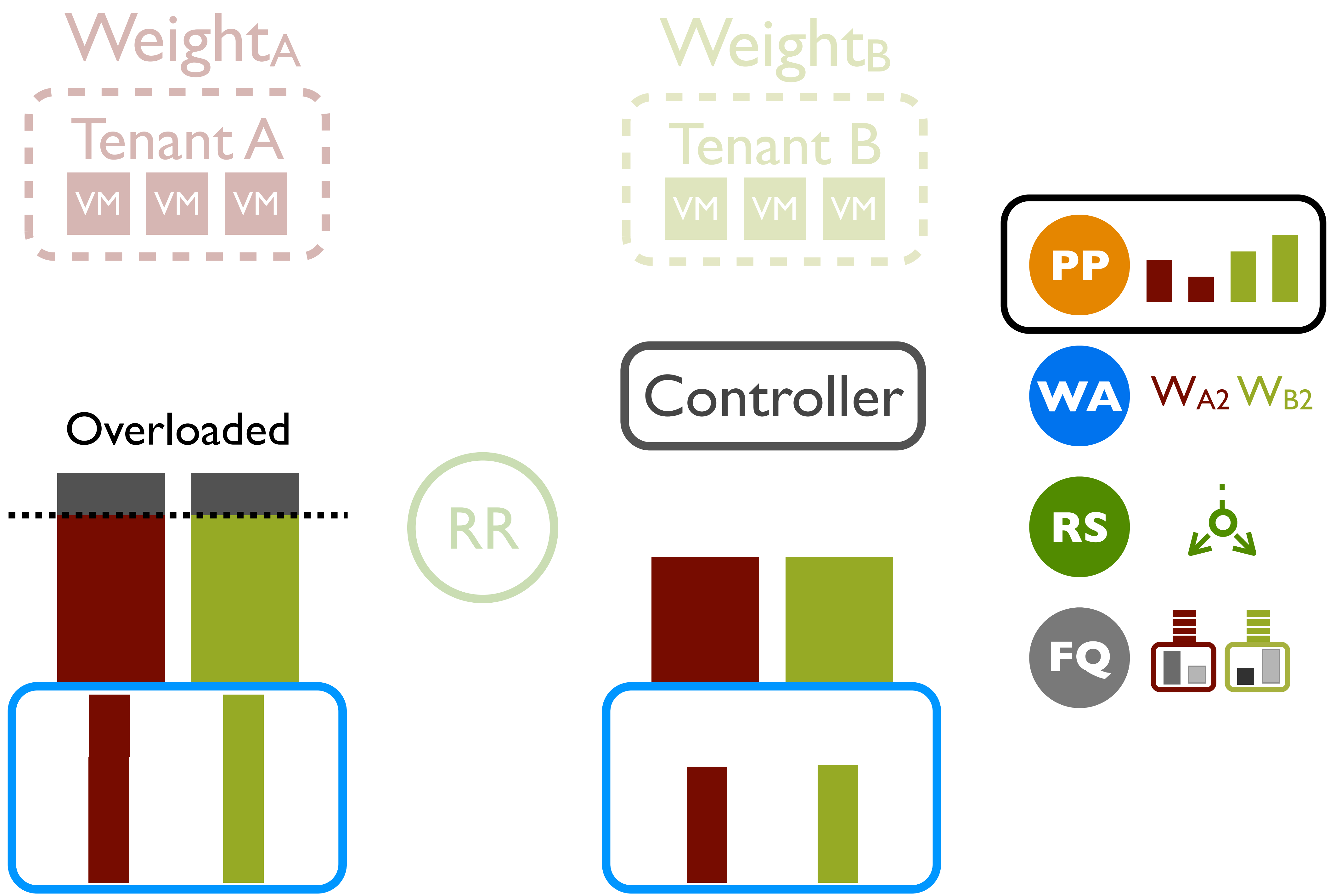
Predictable Multi-Tenant Key-Value Storage



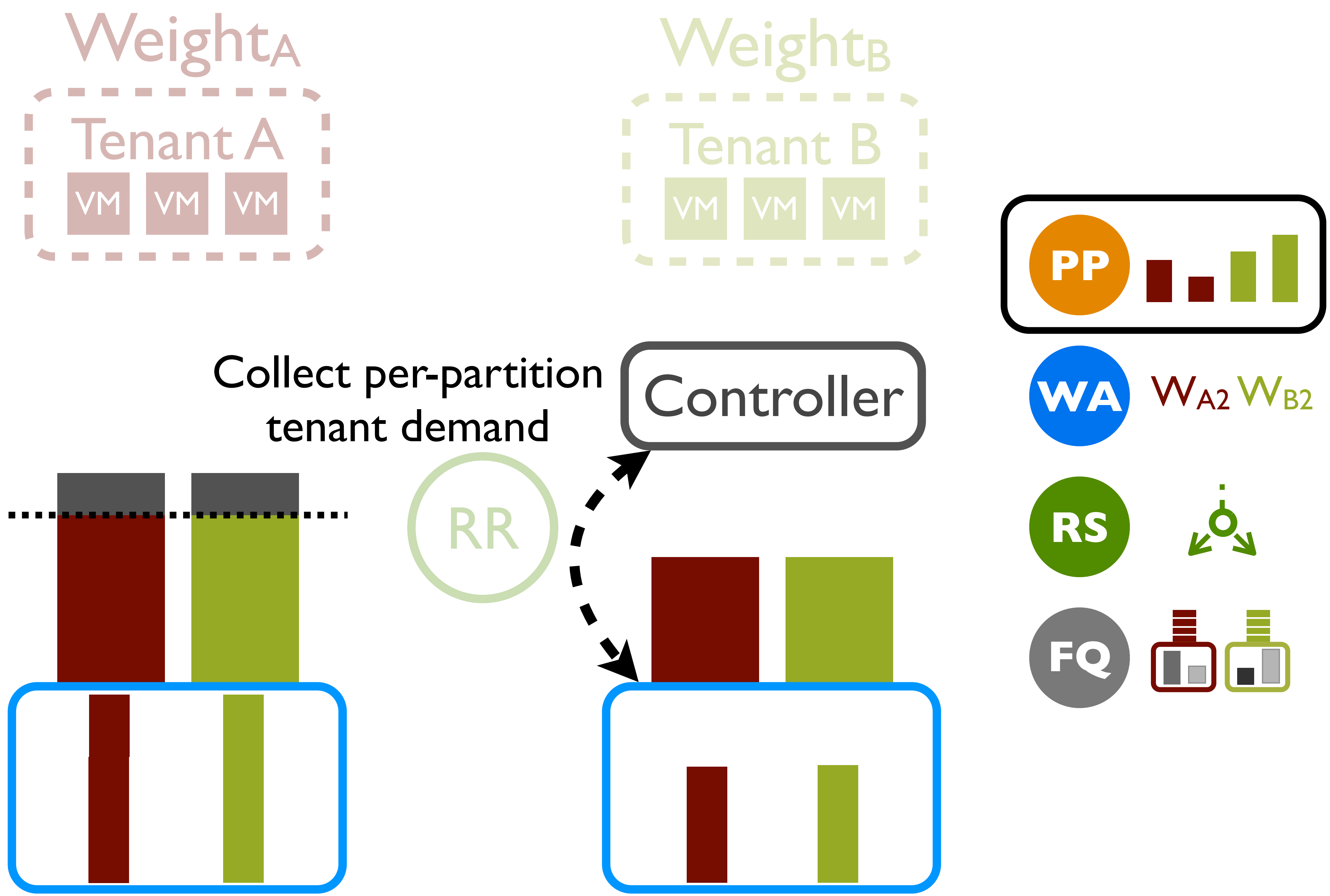
Strawman: Place Partitions Randomly



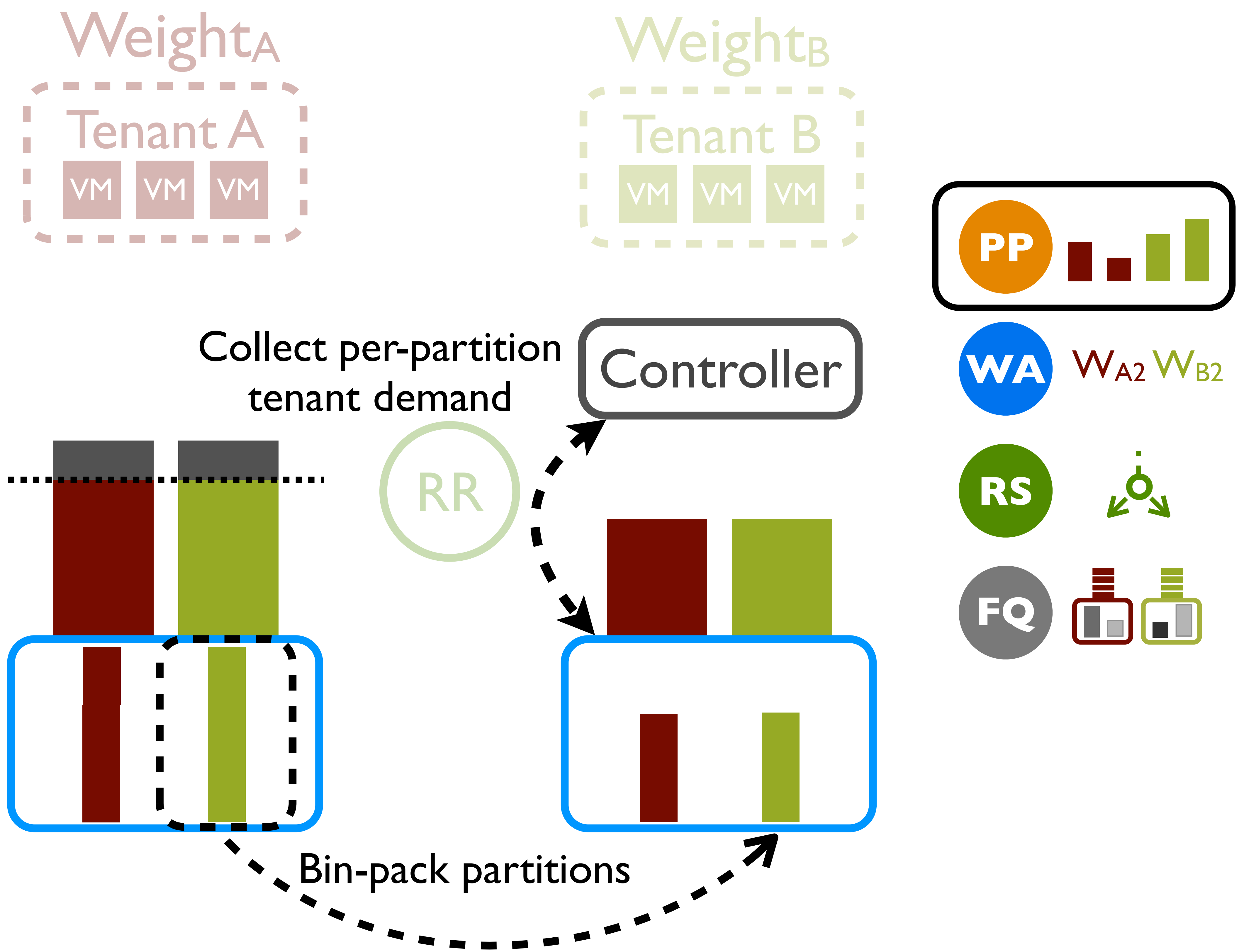
Strawman: Place Partitions Randomly



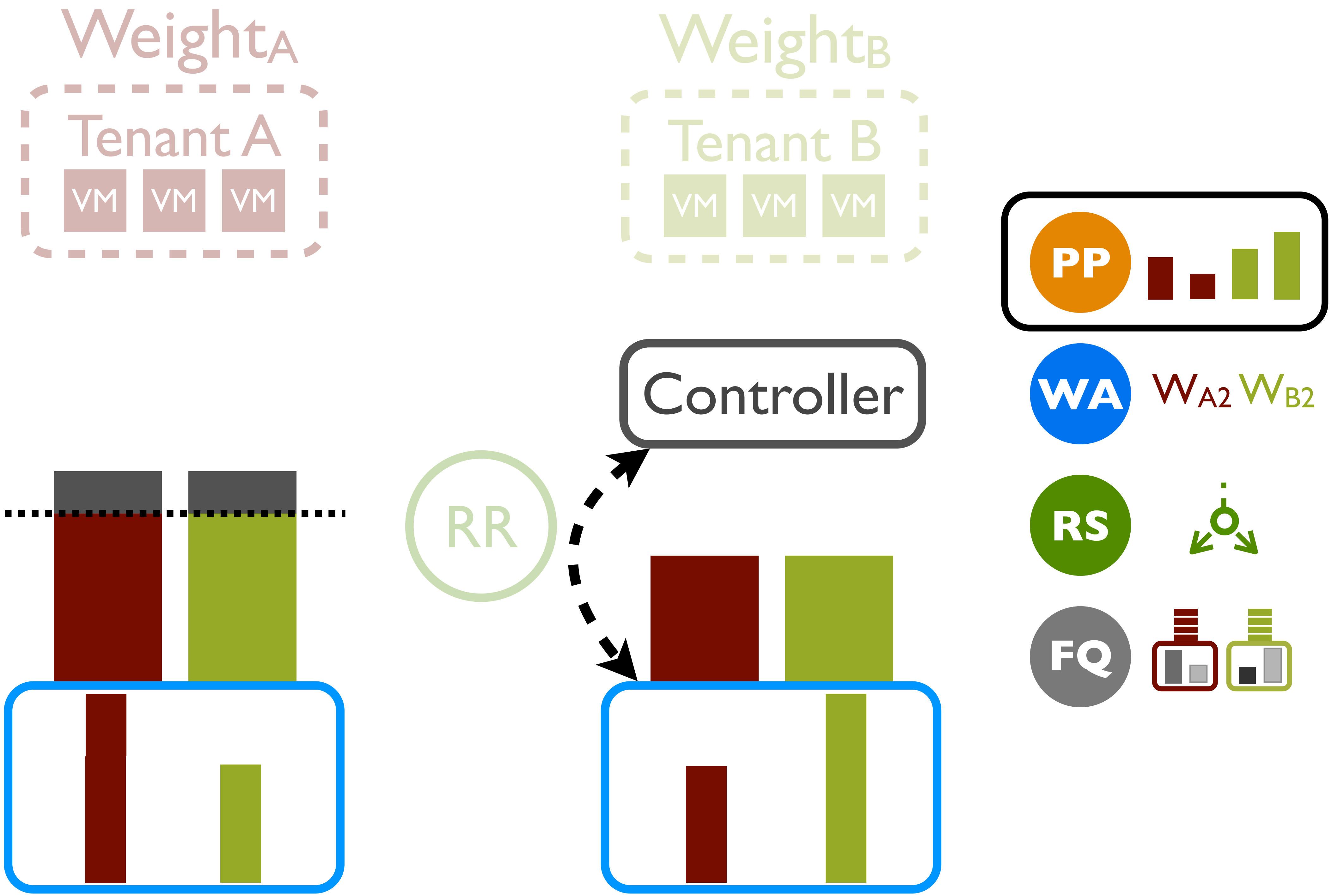
Pisces: Place Partitions By Fairness Constraints



Pisces: Place Partitions By Fairness Constraints

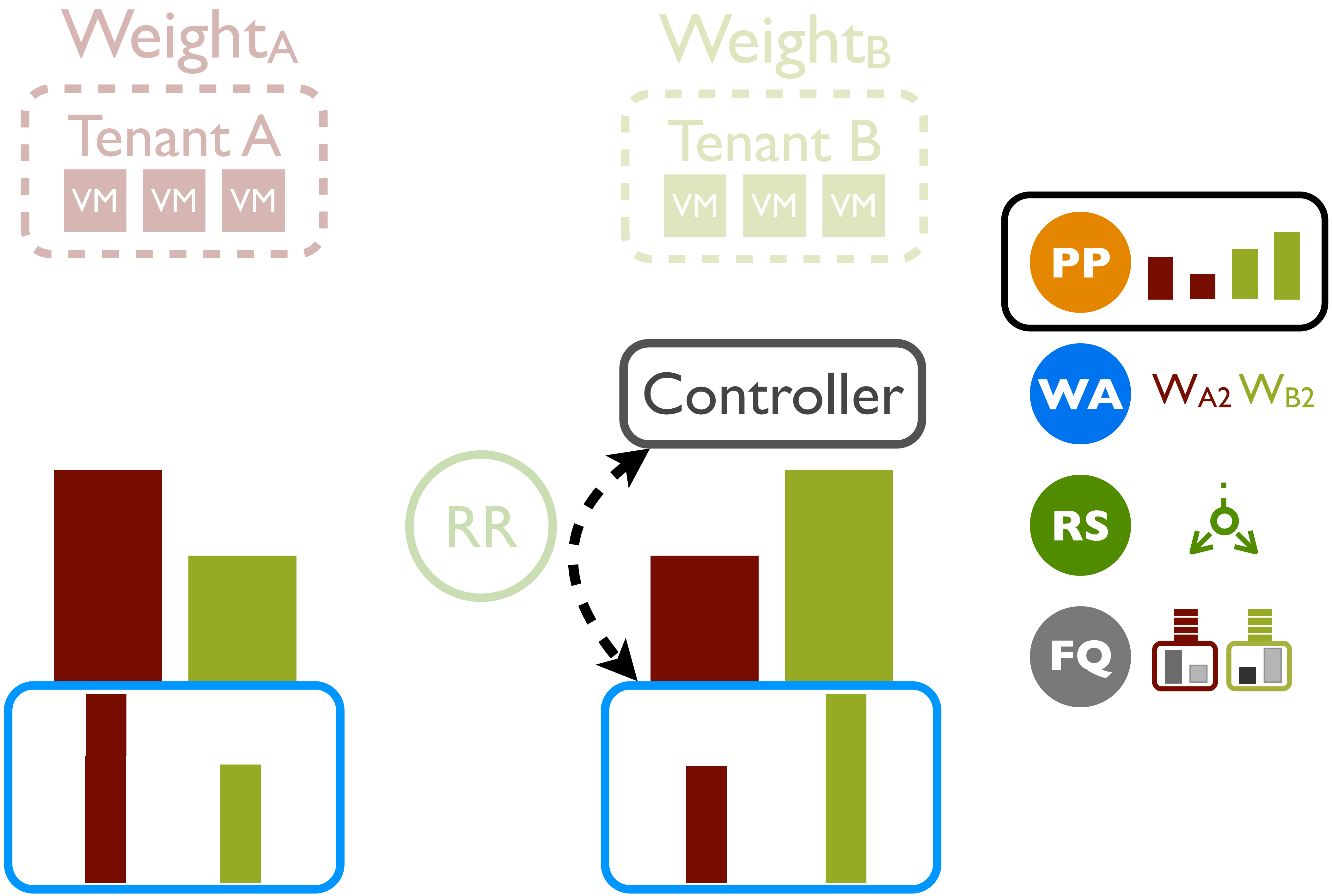


Pisces: Place Partitions By Fairness Constraints



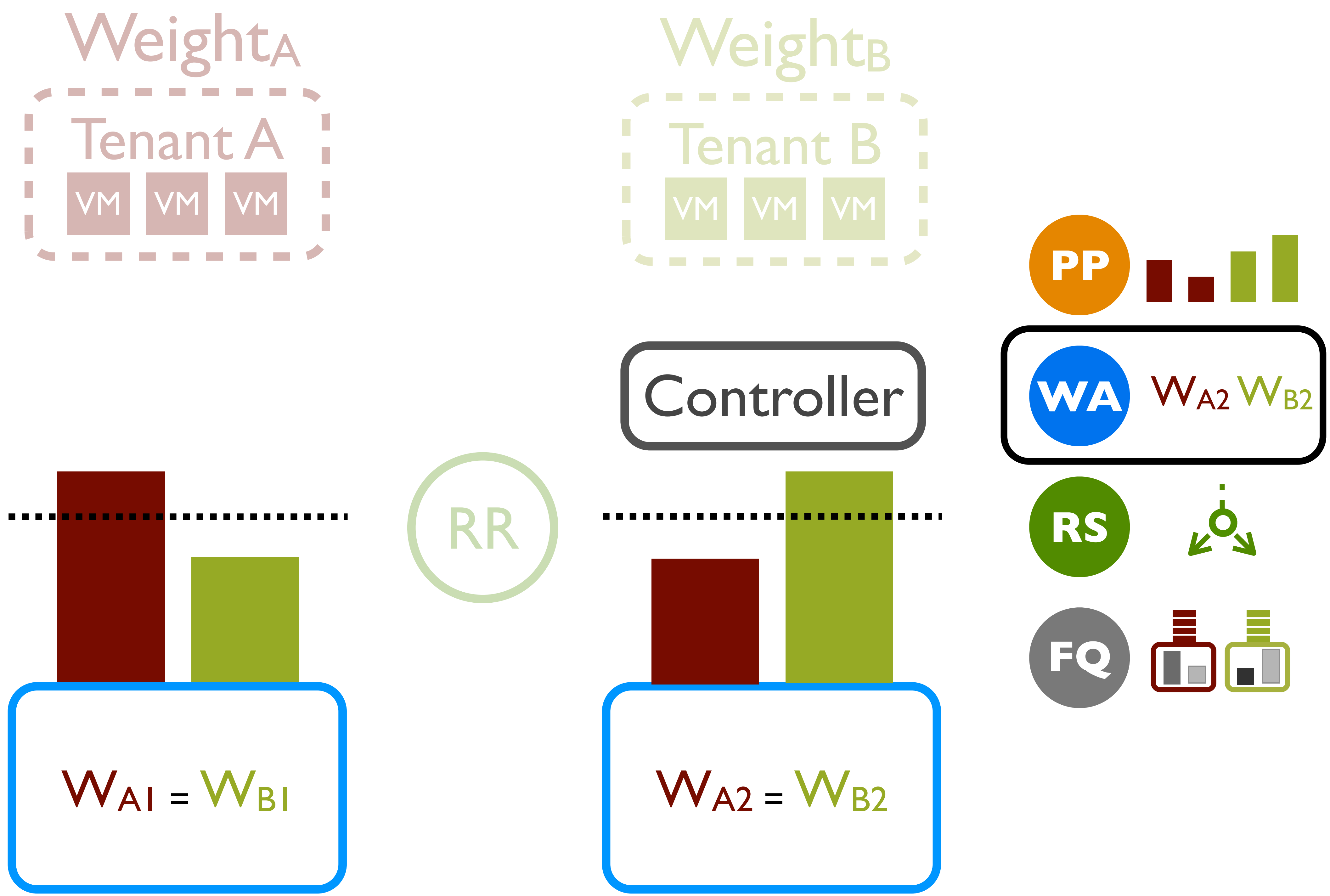
Results in feasible partition placement

Pisces: Place Partitions By Fairness Constraints

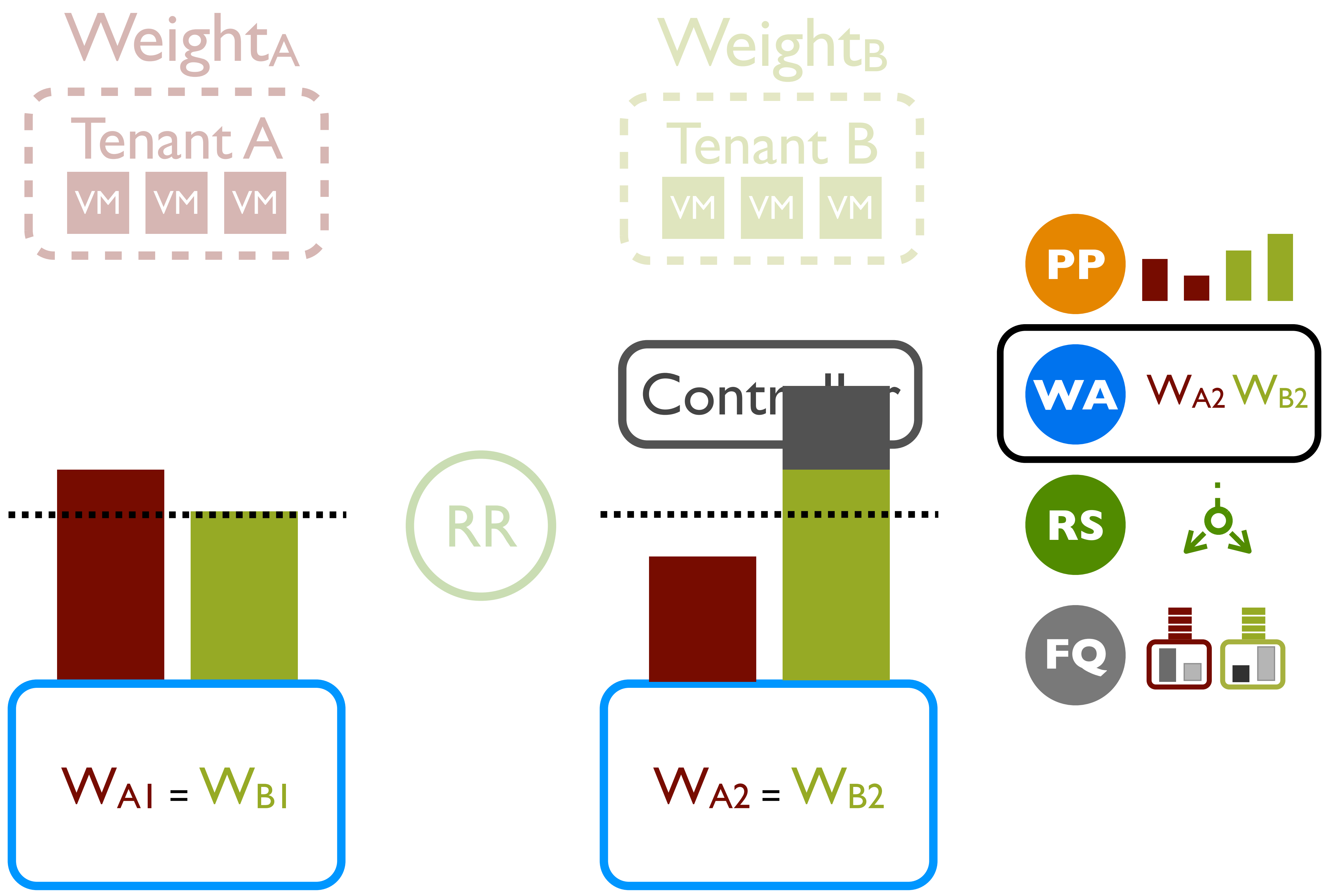


Results in feasible partition placement

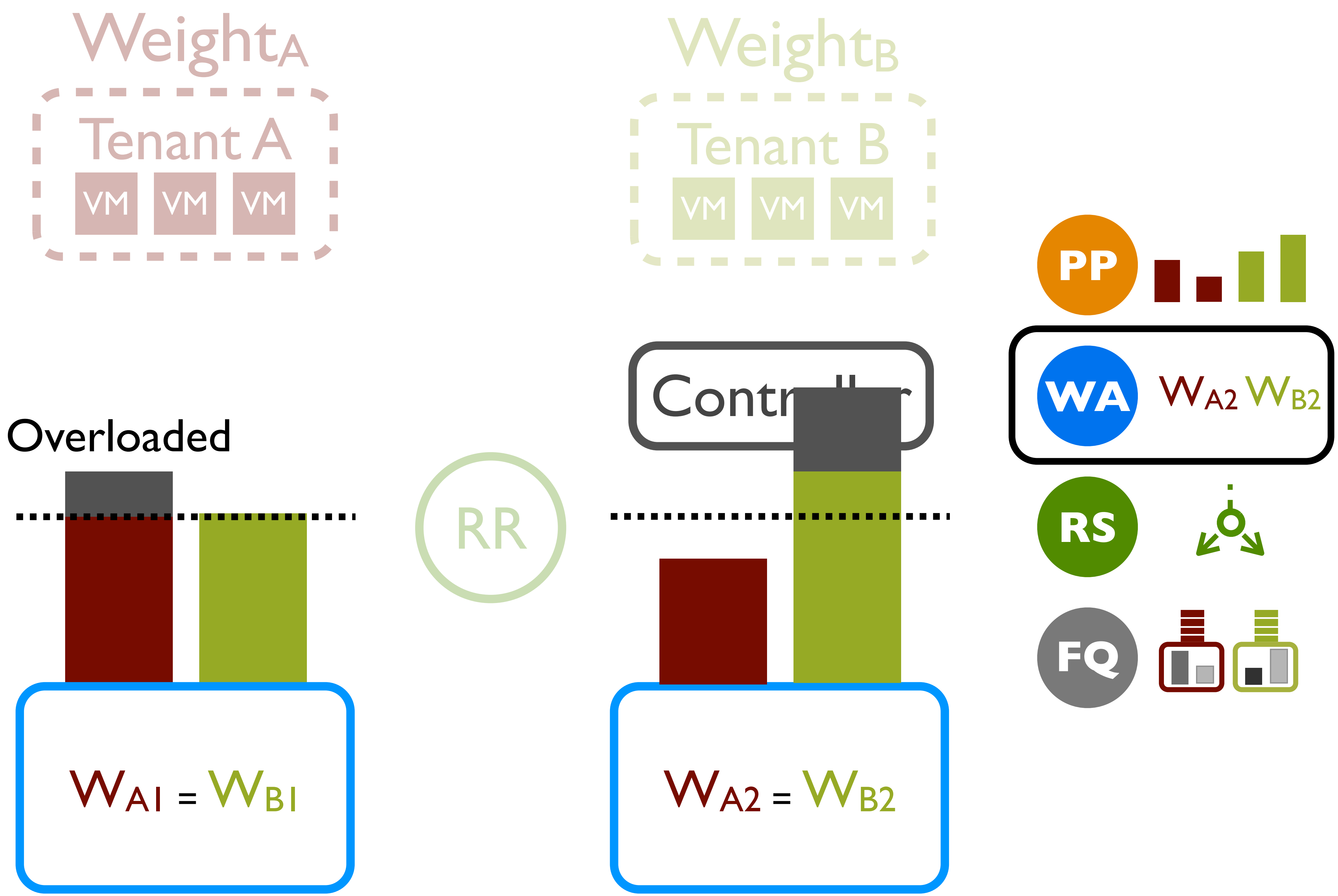
Strawman: Allocate Local Weights Evenly



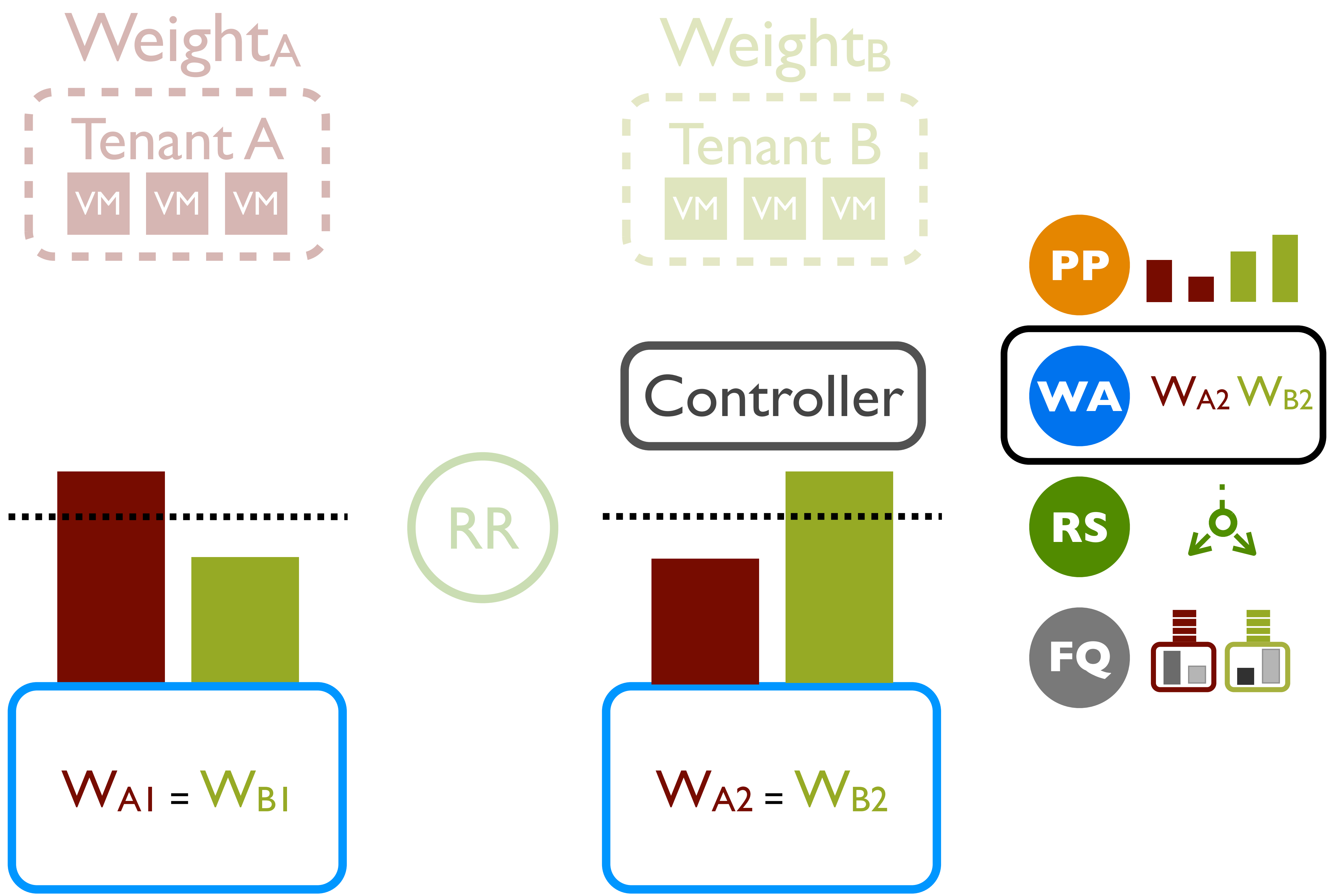
Strawman: Allocate Local Weights Evenly



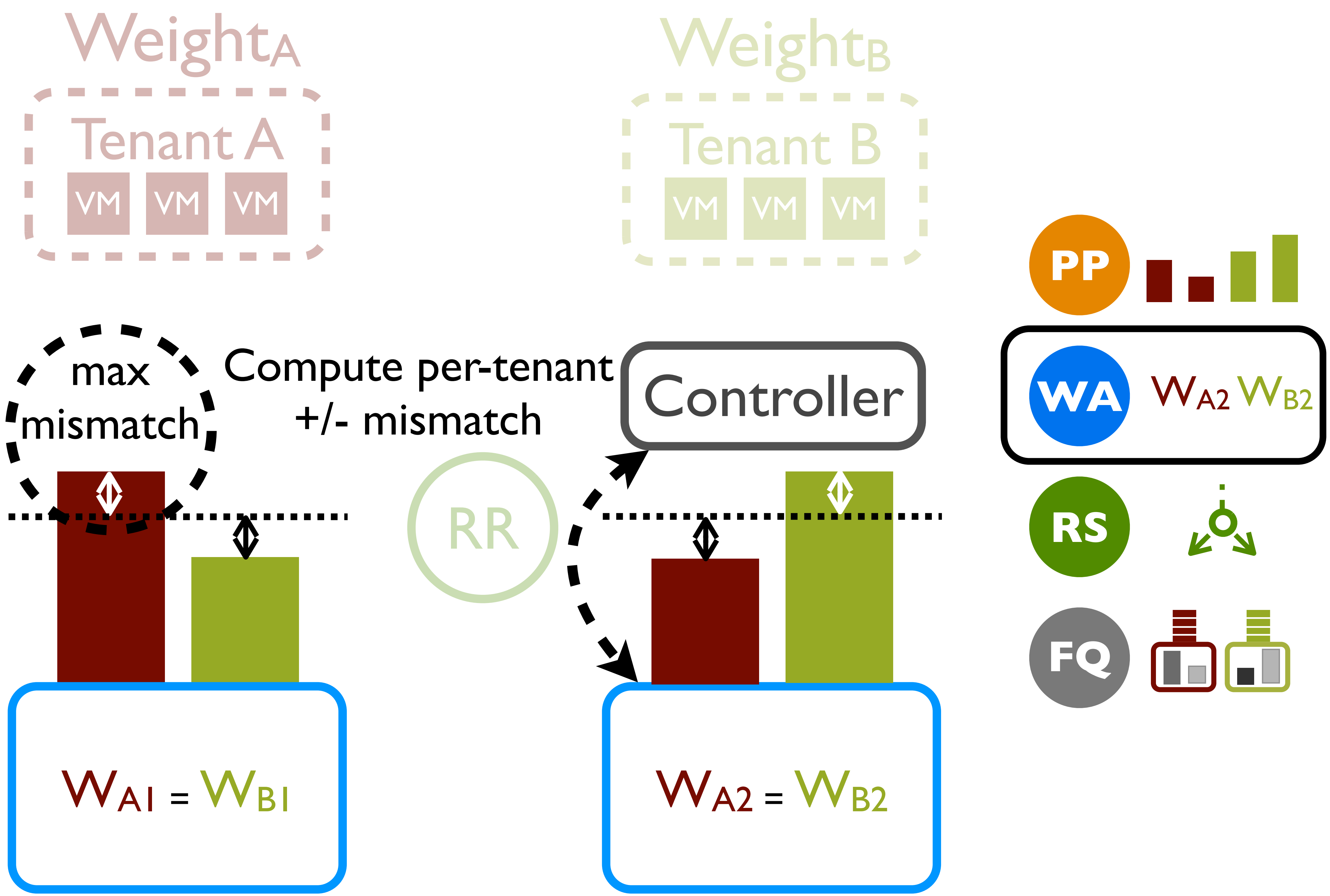
Strawman: Allocate Local Weights Evenly



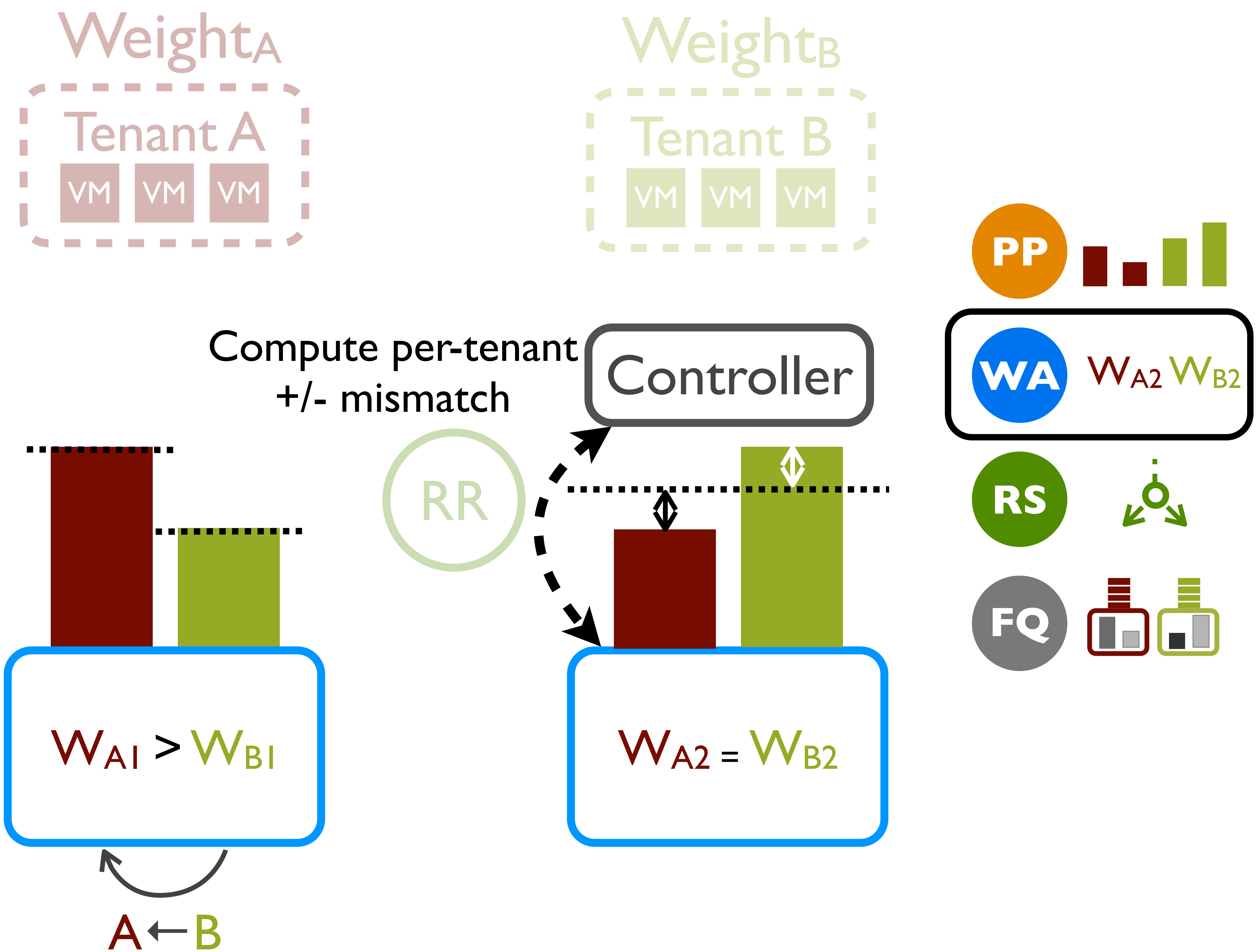
Pisces: Allocate Local Weights By Tenant Demand



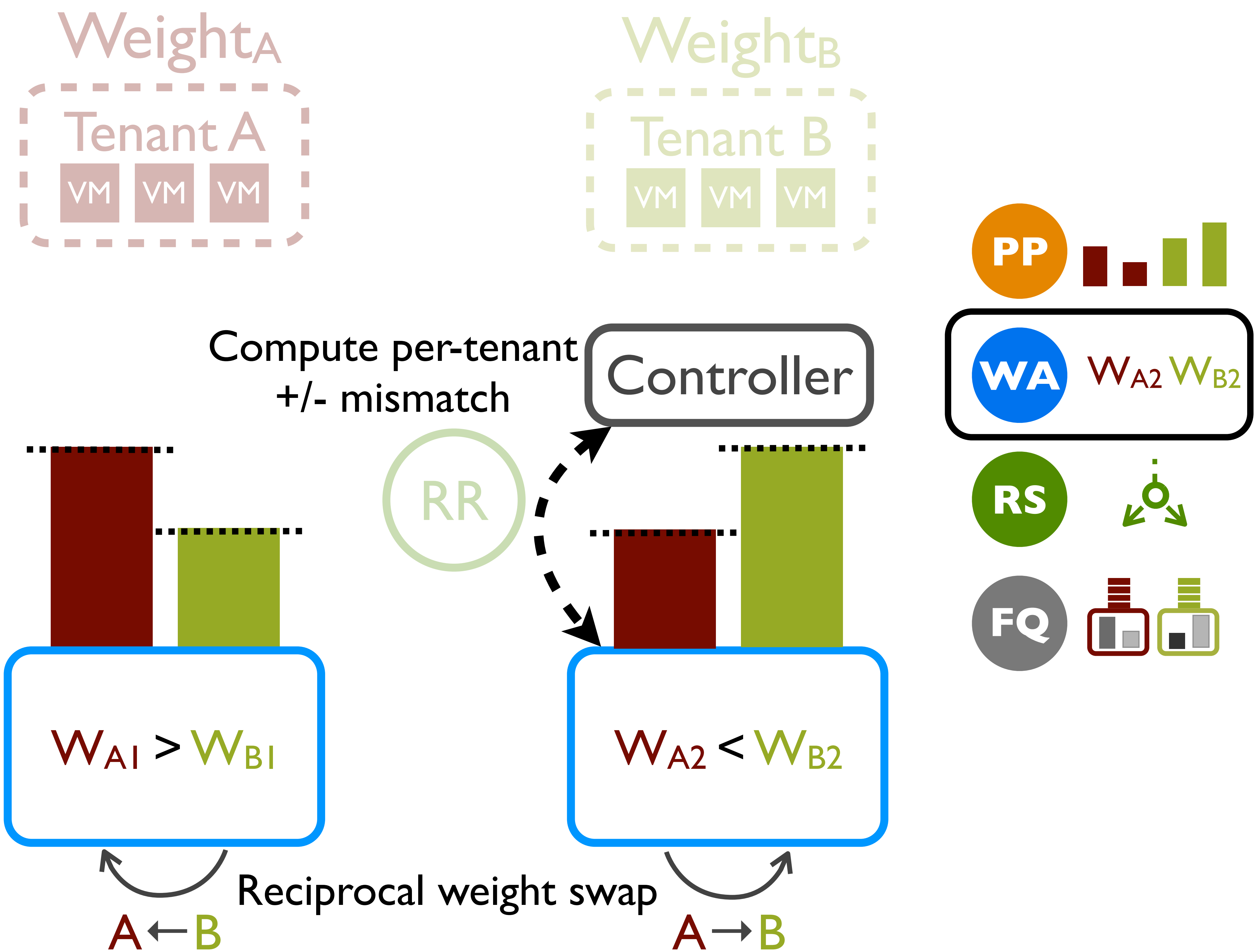
Pisces: Allocate Local Weights By Tenant Demand



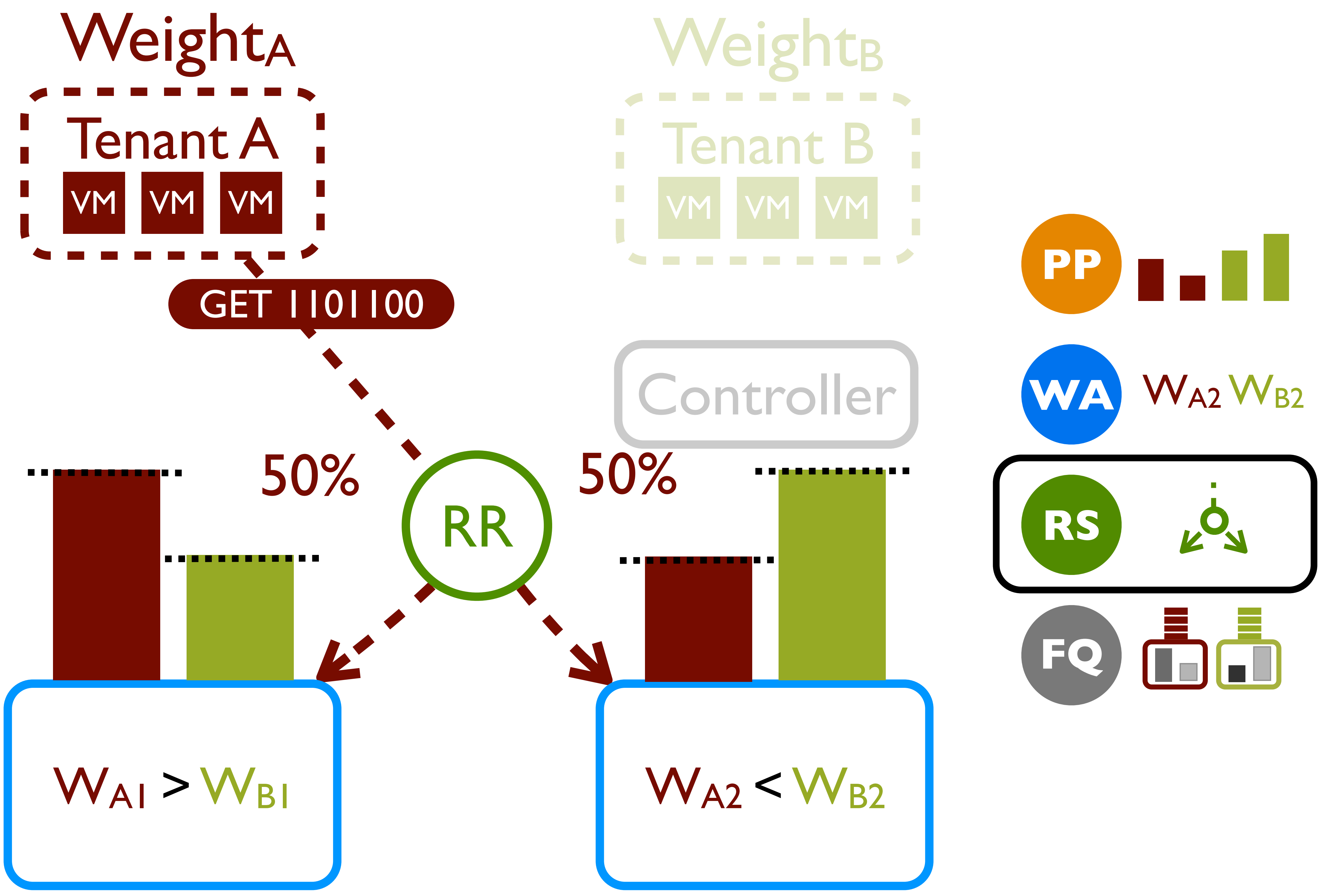
Pisces: Allocate Local Weights By Tenant Demand



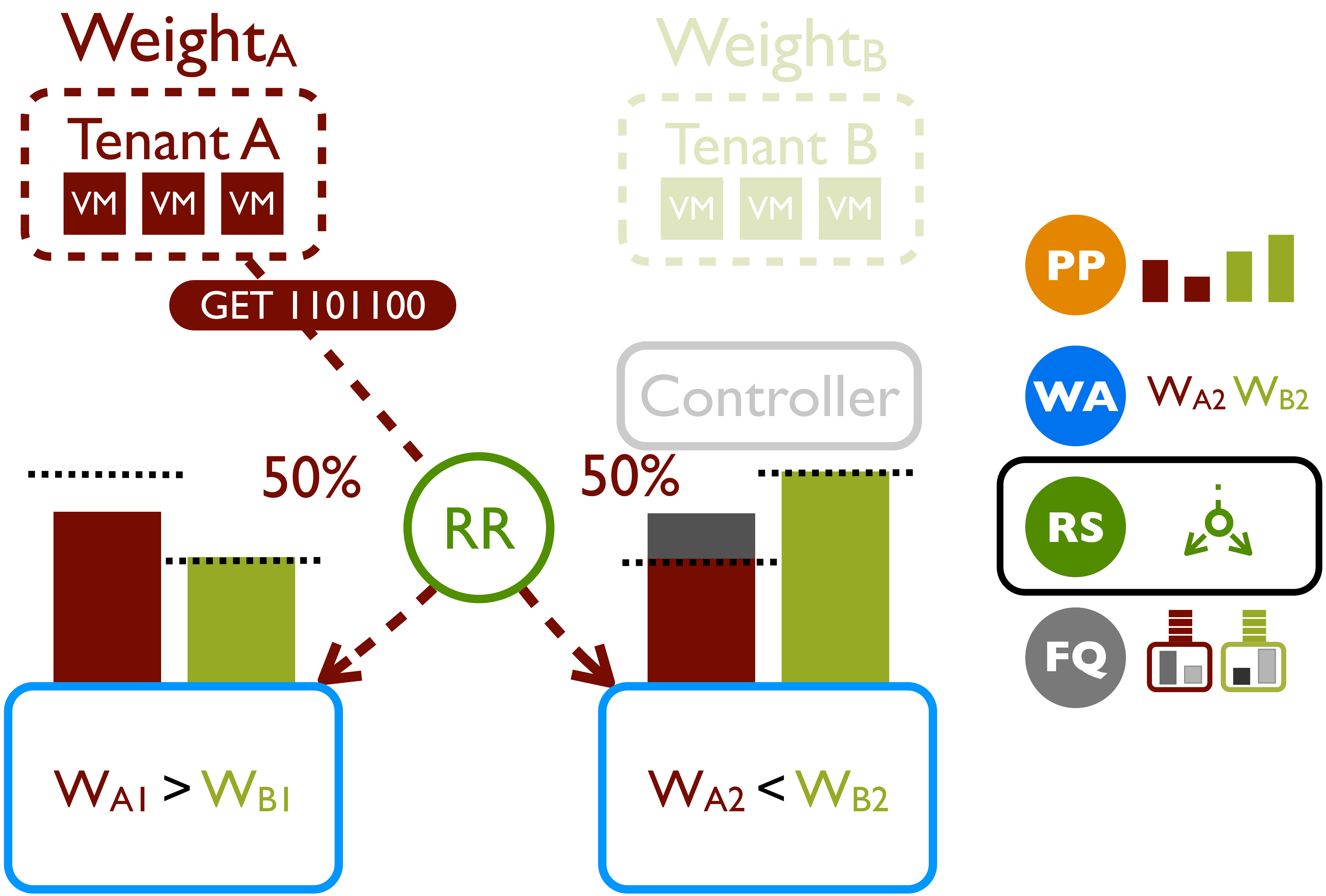
Pisces: Allocate Local Weights By Tenant Demand



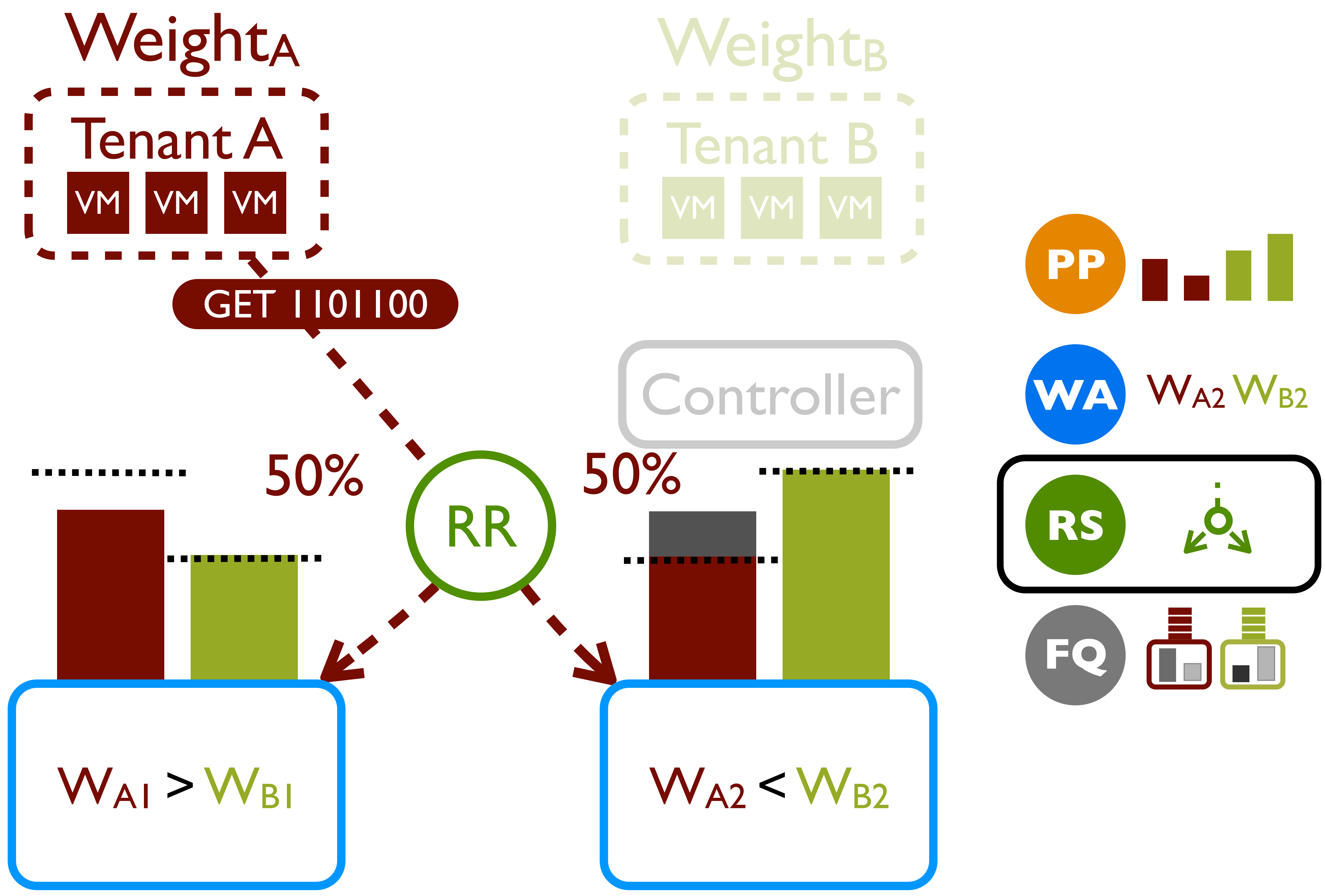
Strawman: Select Replicas Evenly



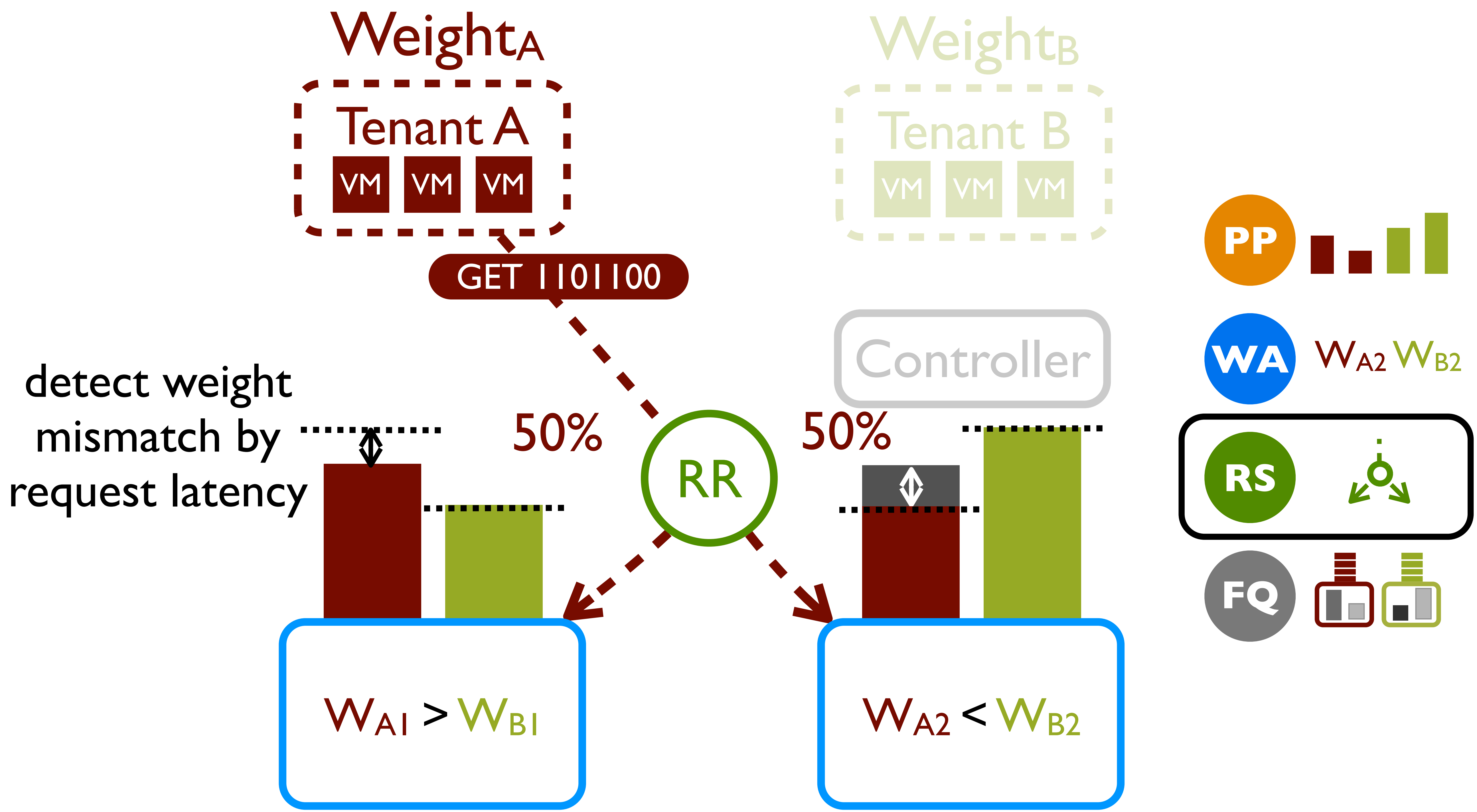
Strawman: Select Replicas Evenly



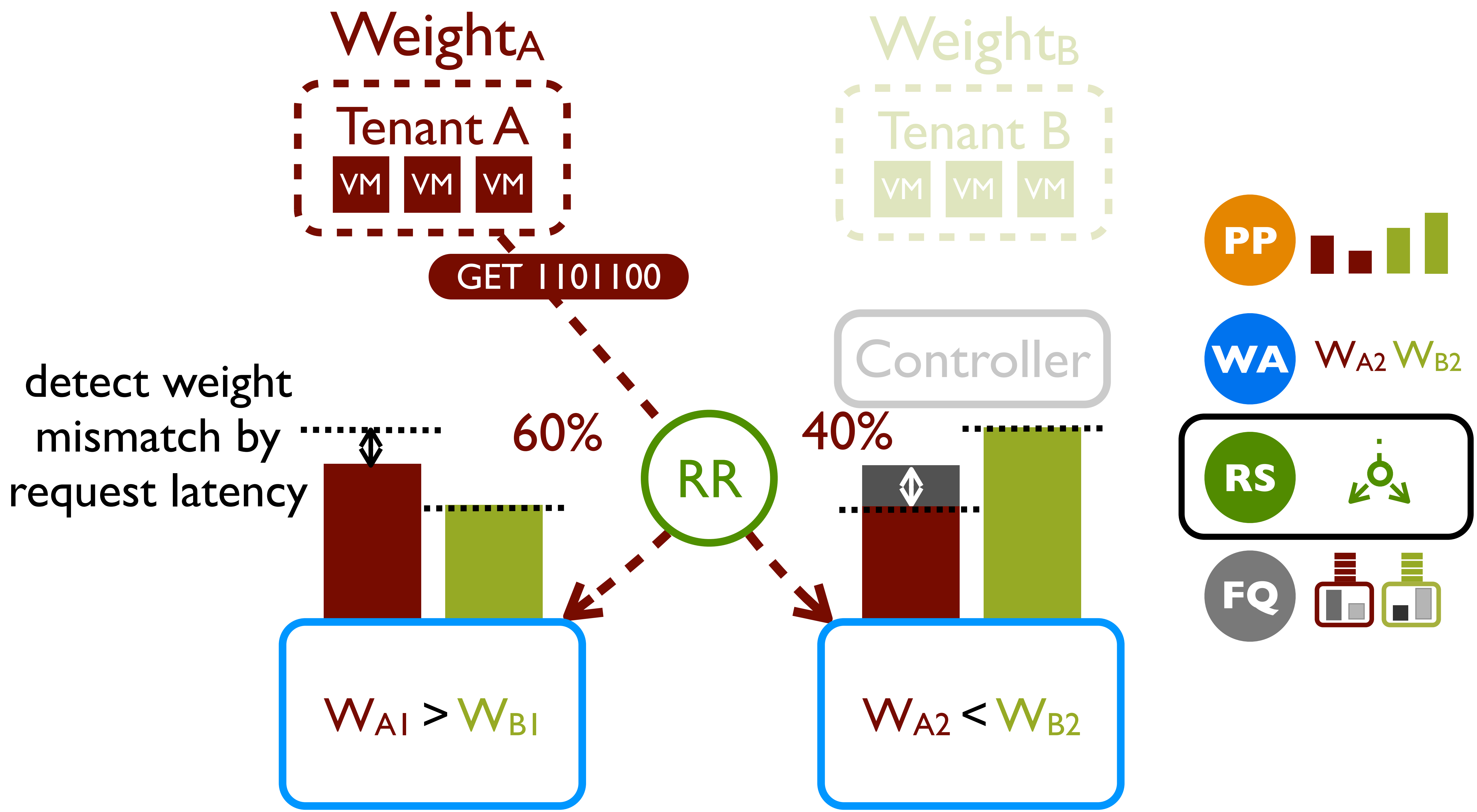
Pisces: Select Replicas By Local Weight



Pisces: Select Replicas By Local Weight

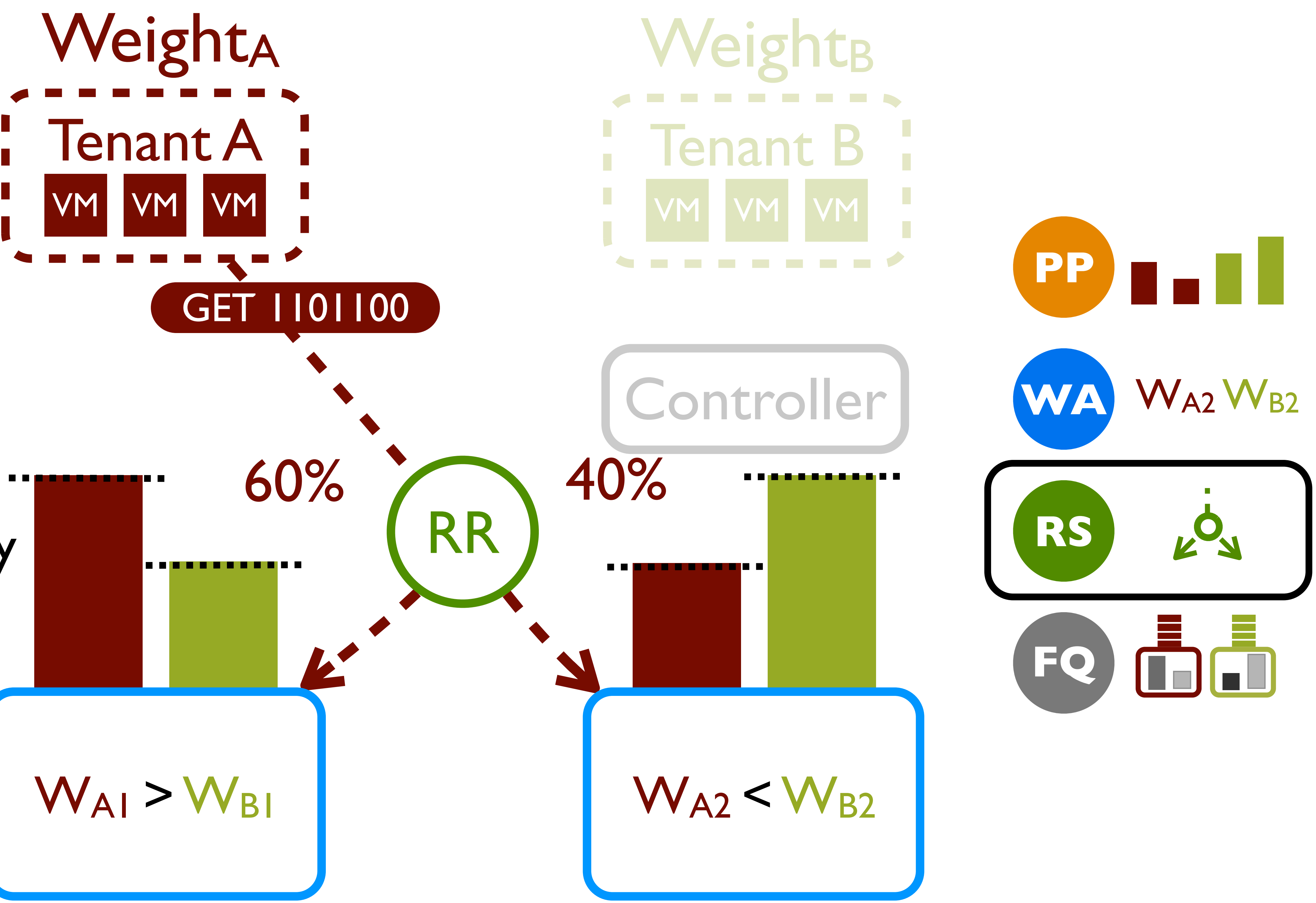


Pisces: Select Replicas By Local Weight

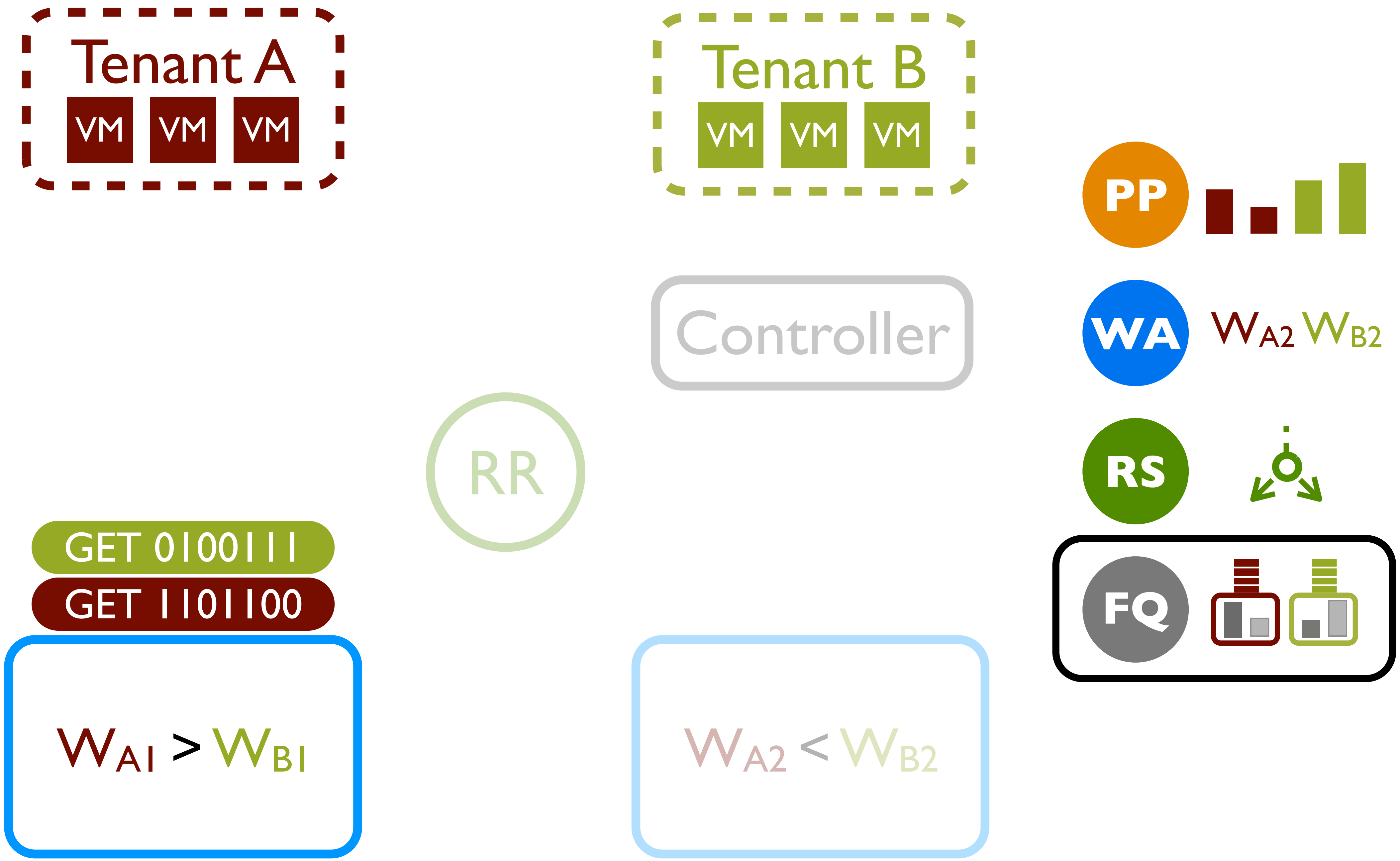


Pisces: Select Replicas By Local Weight

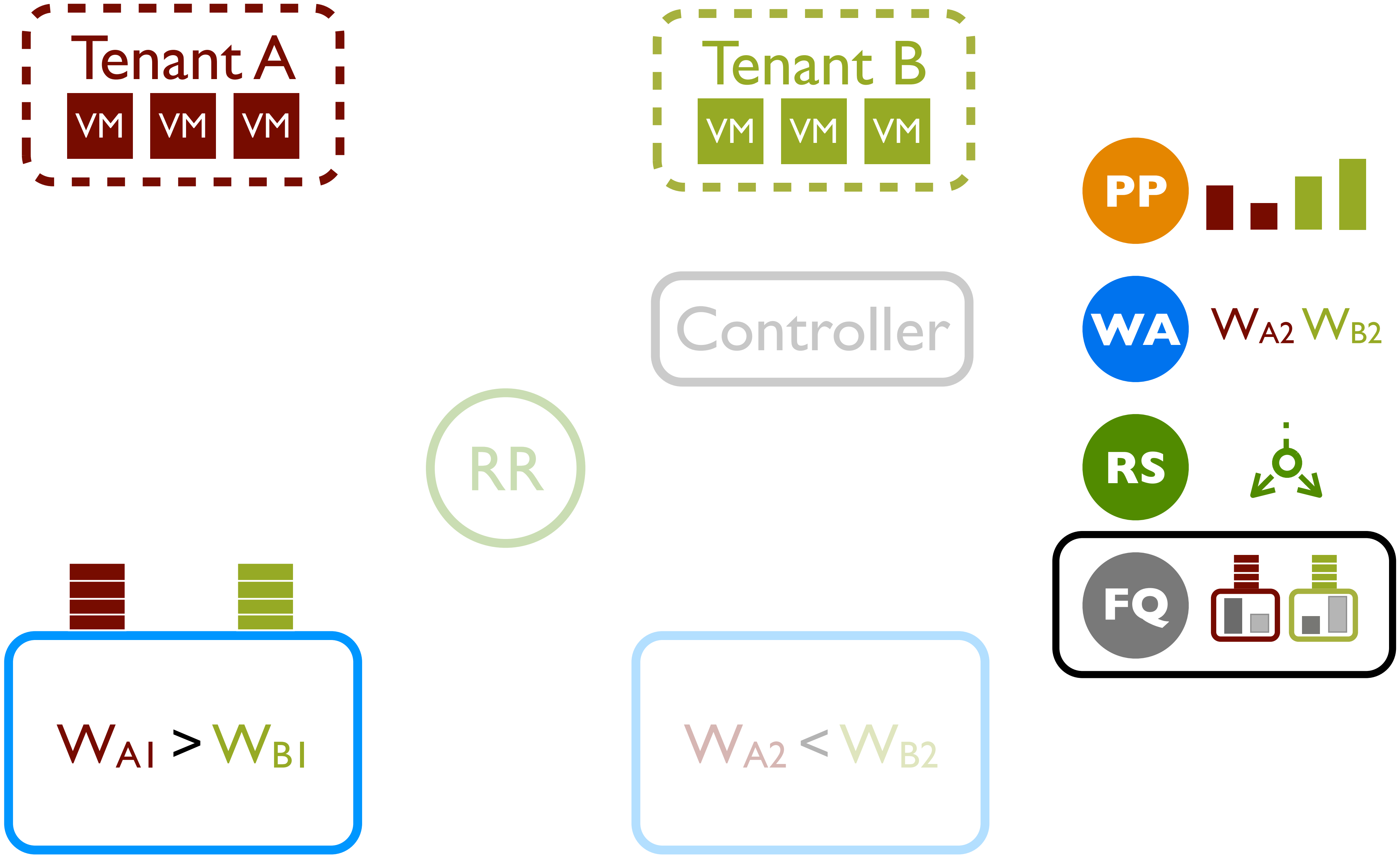
detect weight mismatch by request latency



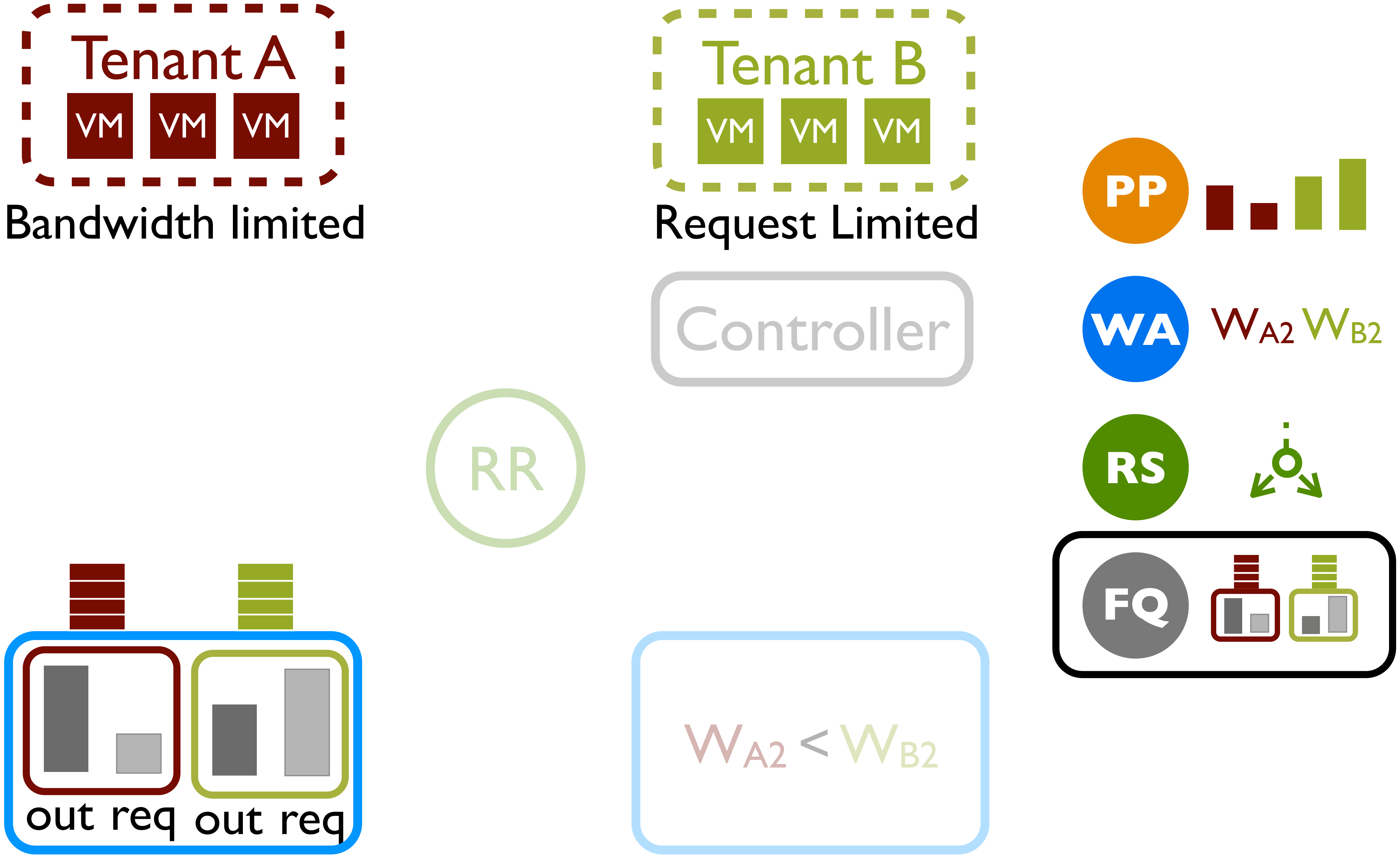
Strawman: Queue Tenants By Single Resource



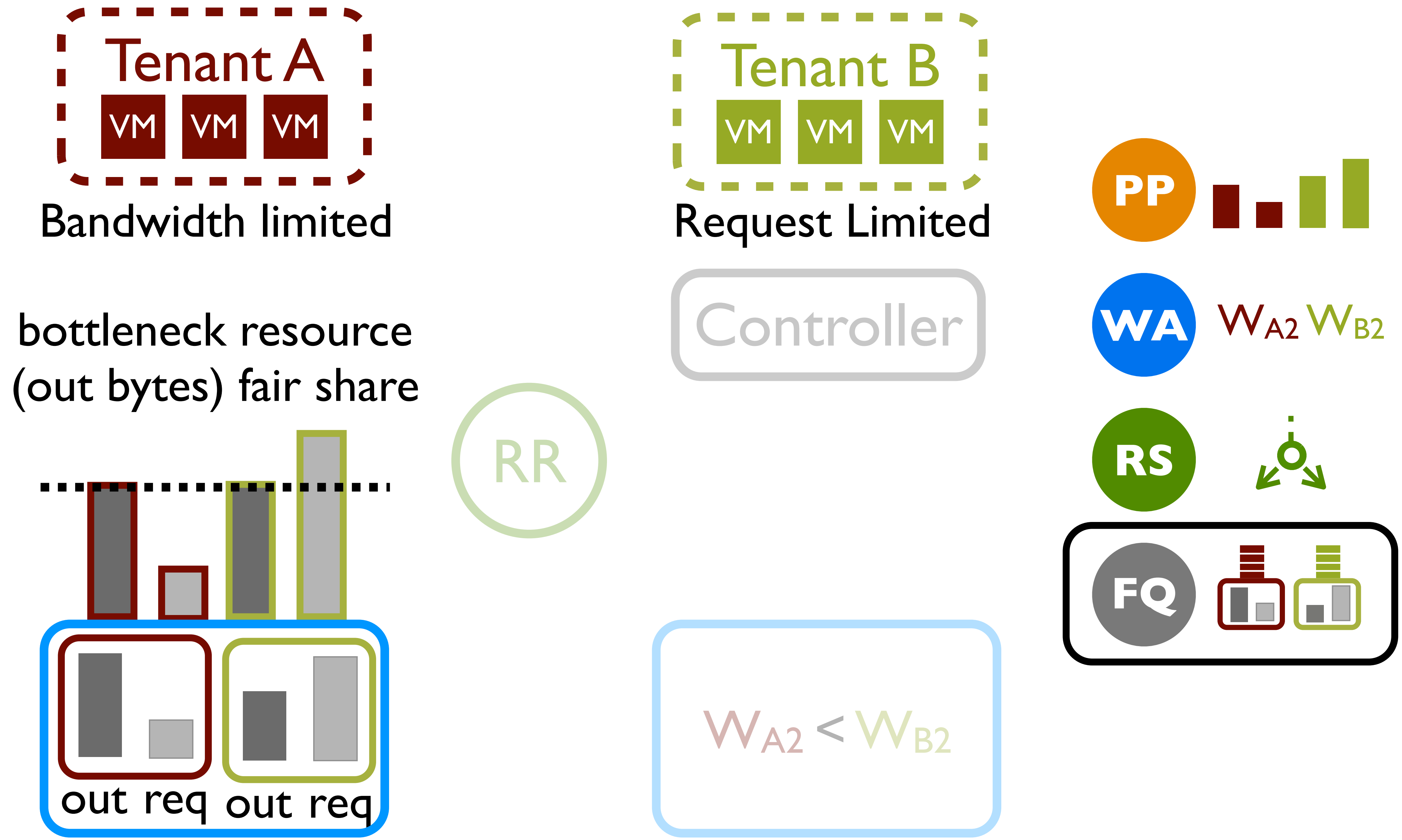
Strawman: Queue Tenants By Single Resource



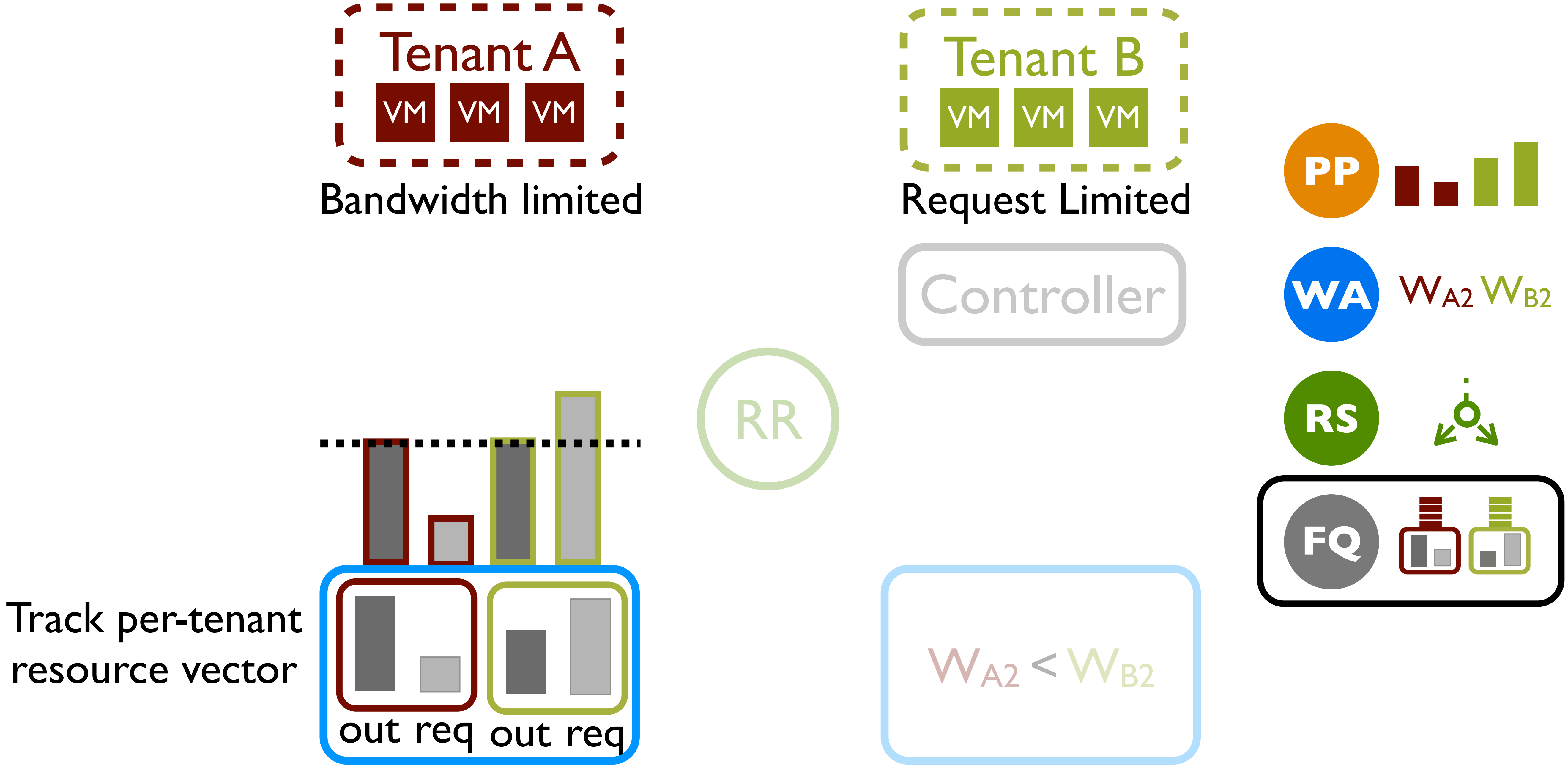
Strawman: Queue Tenants By Single Resource



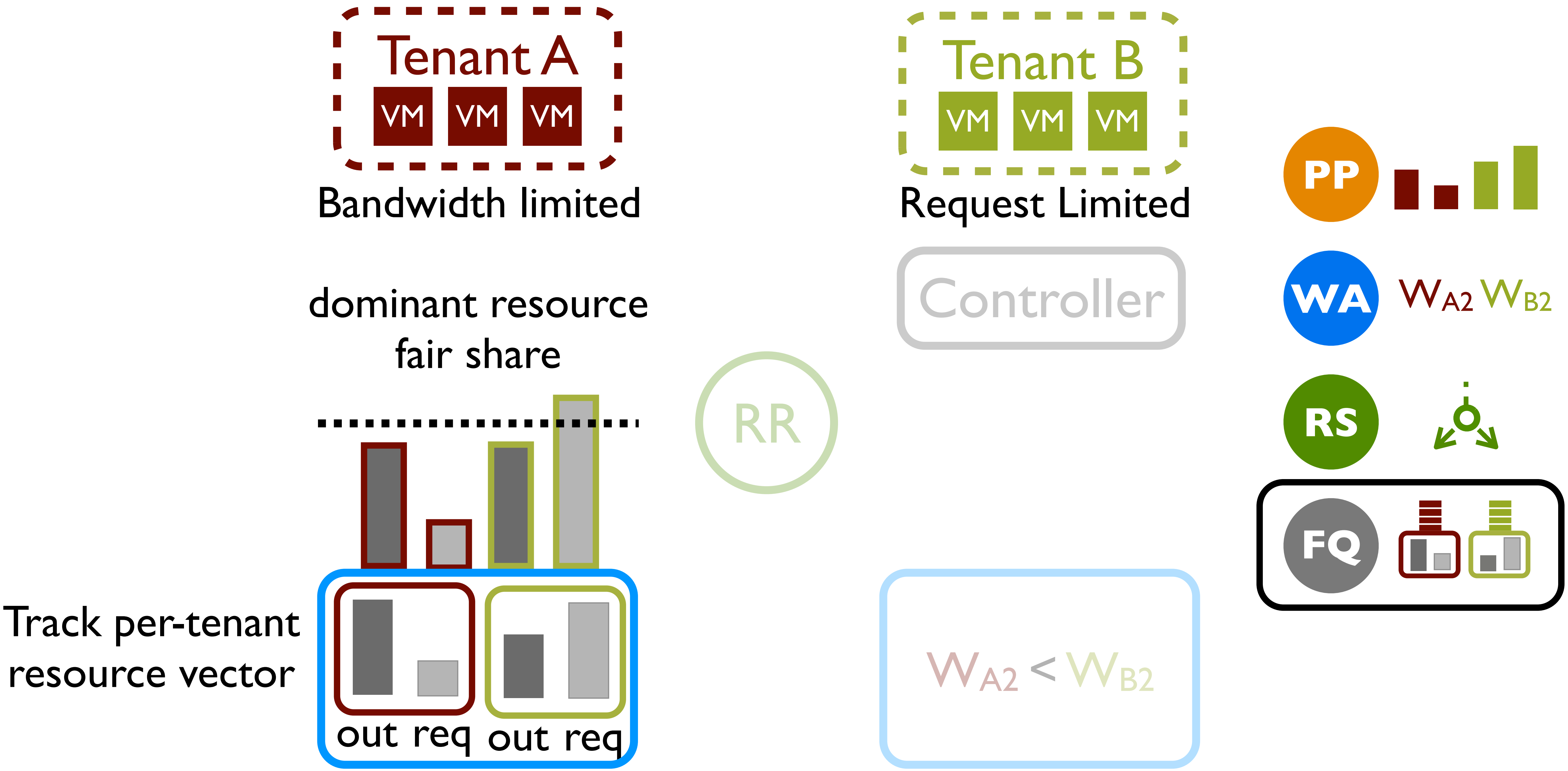
Strawman: Queue Tenants By Single Resource



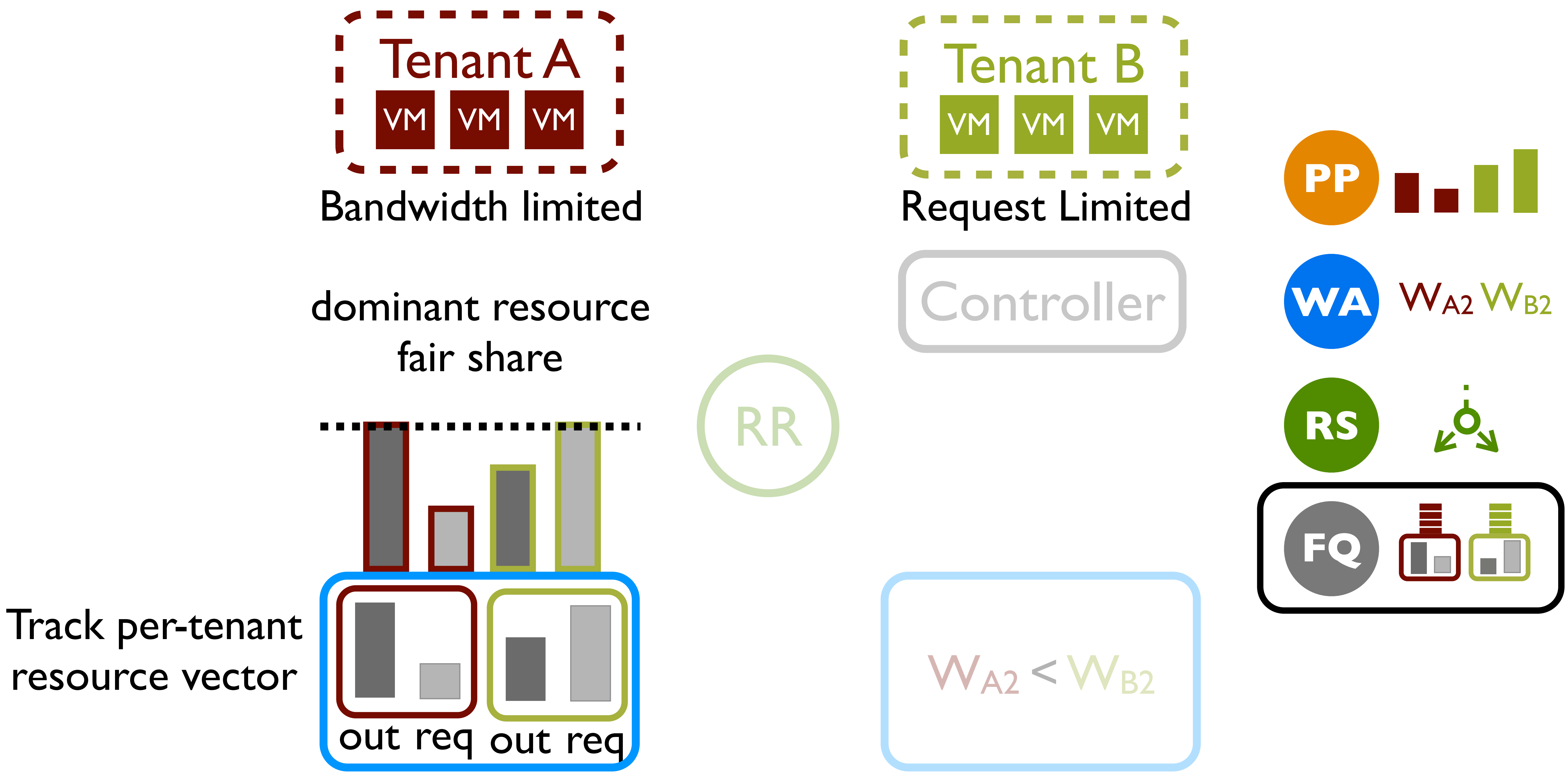
Pisces: Queue Tenants By Dominant Resource



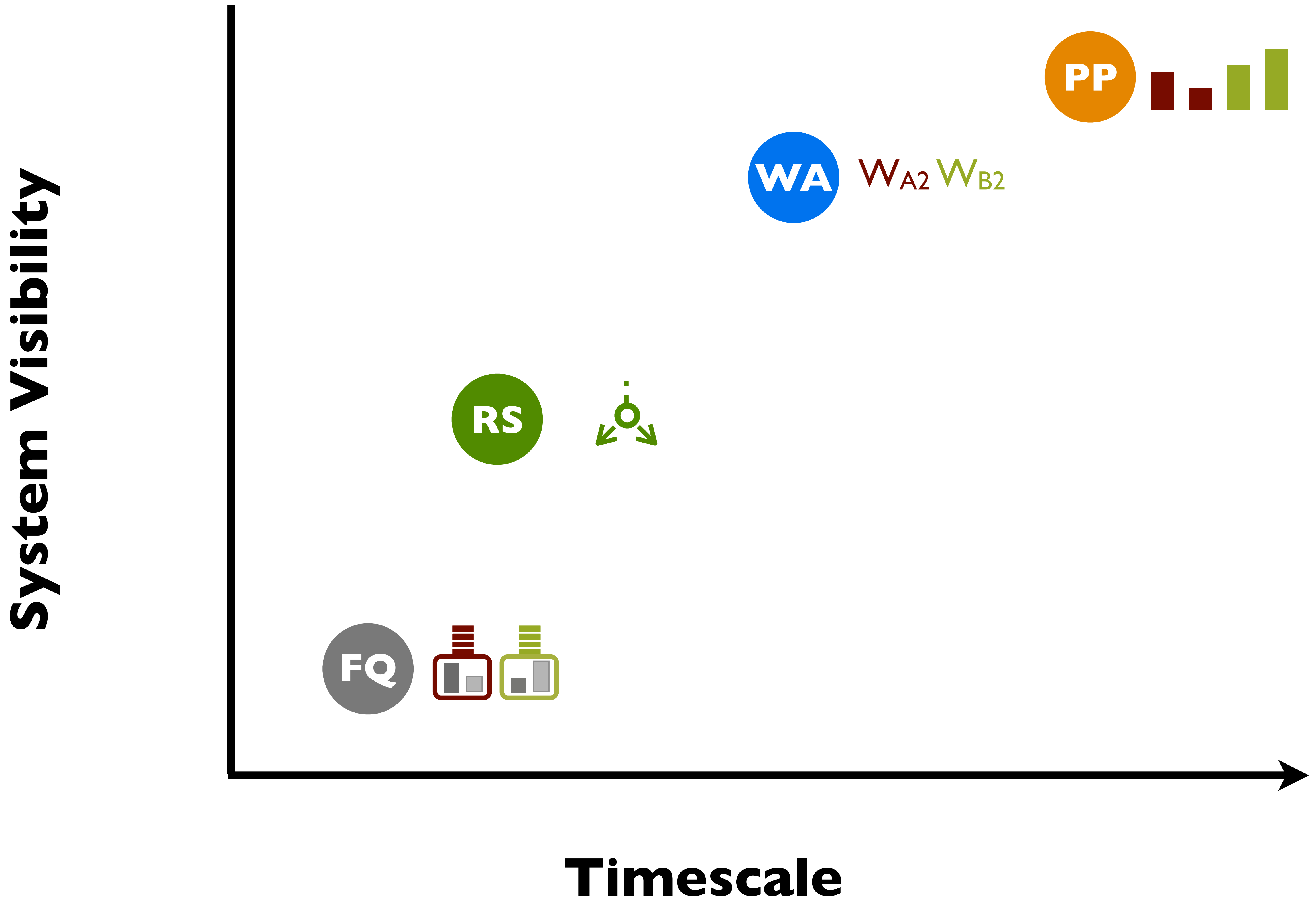
Pisces: Queue Tenants By Dominant Resource



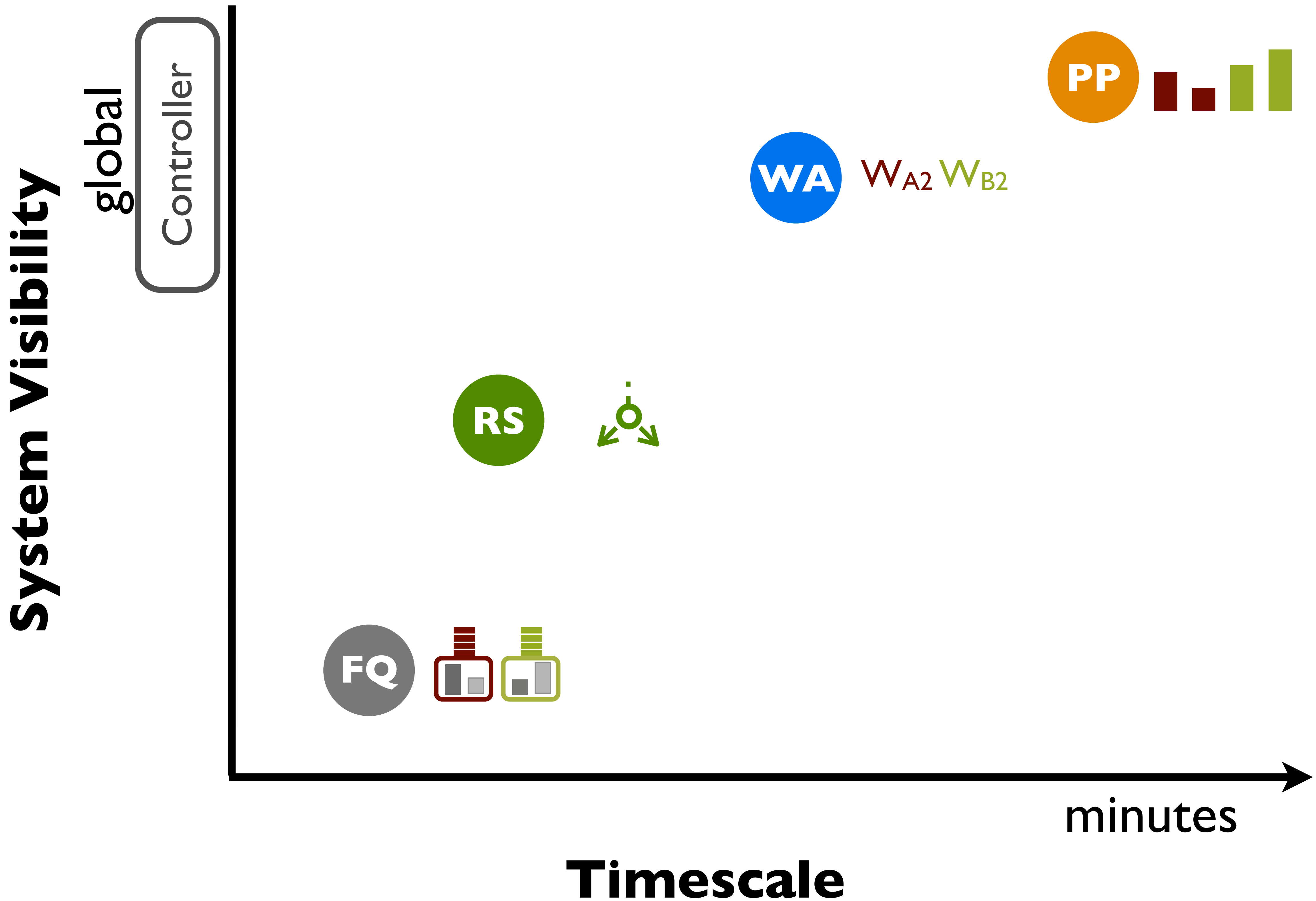
Pisces: Queue Tenants By Dominant Resource



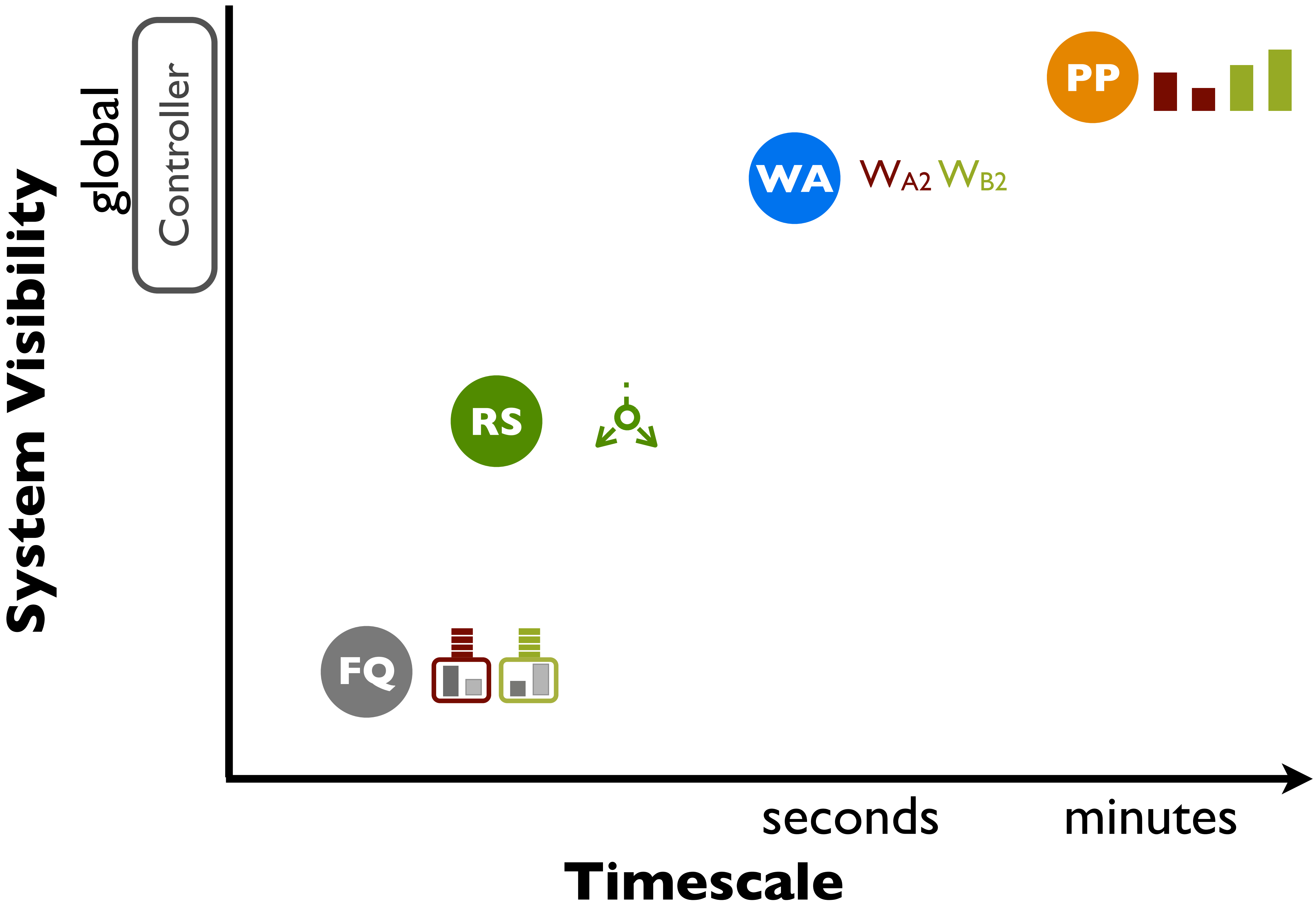
Pisces Mechanisms Solve For Global Fairness



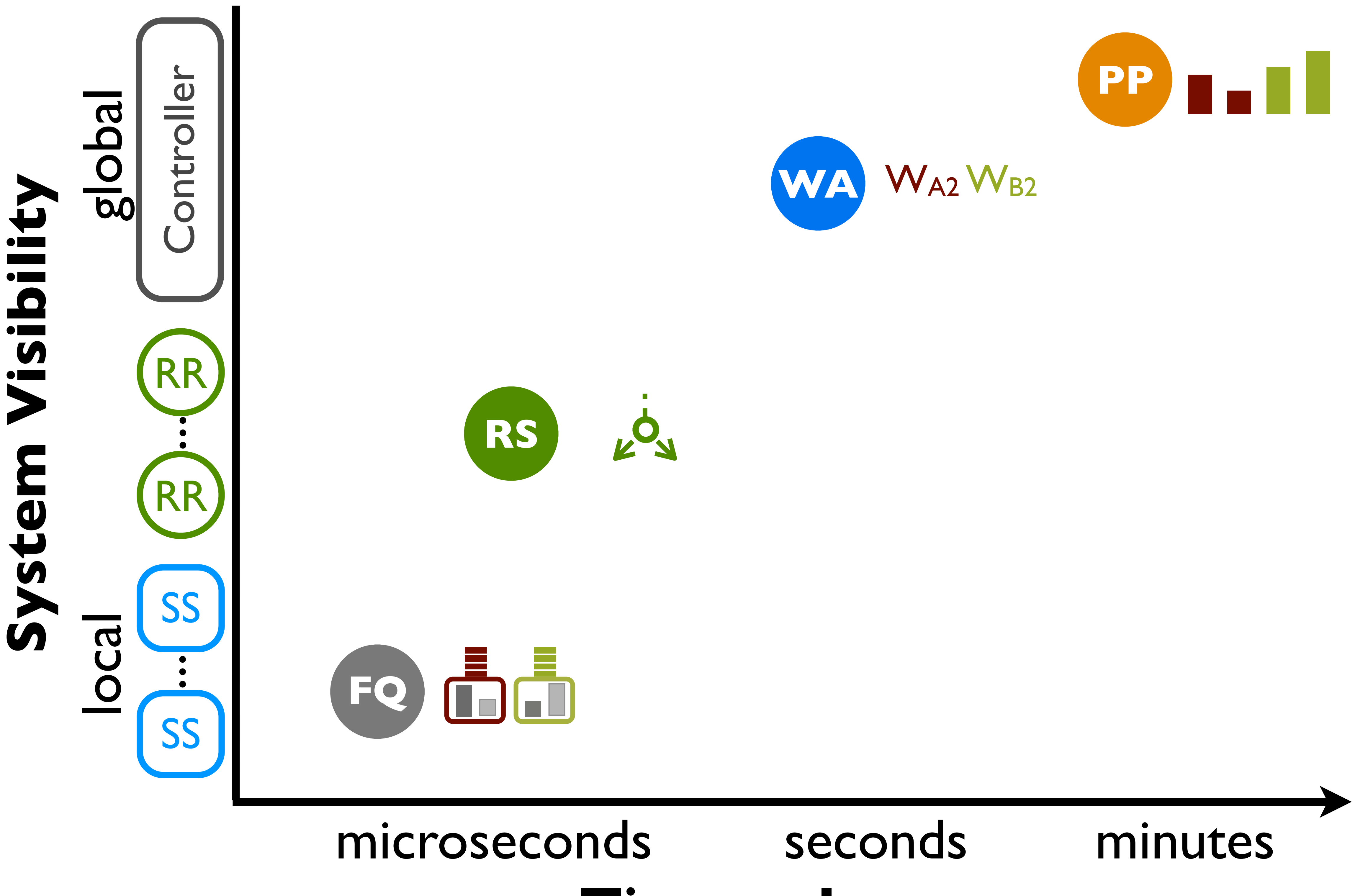
Pisces Mechanisms Solve For Global Fairness



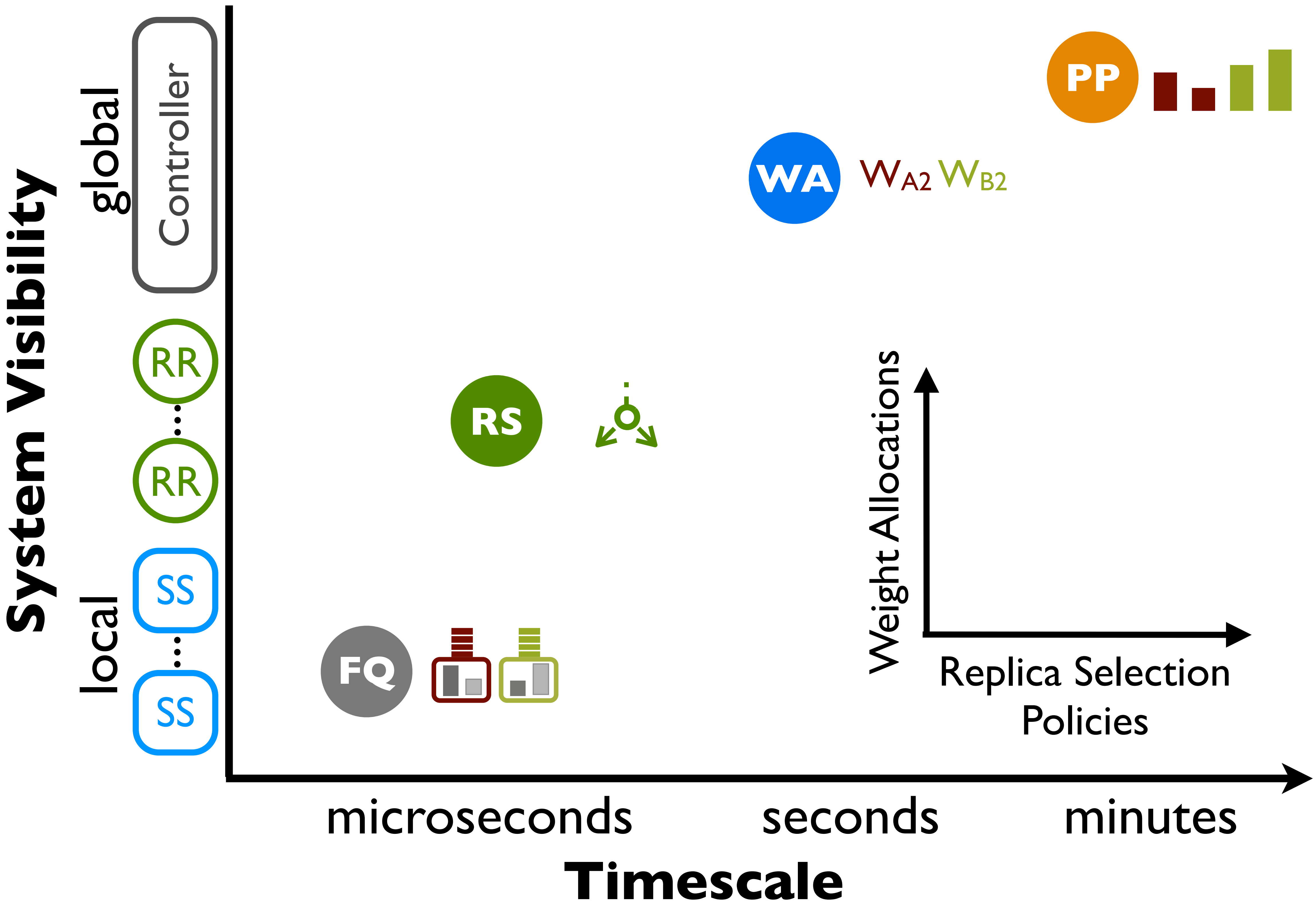
Pisces Mechanisms Solve For Global Fairness



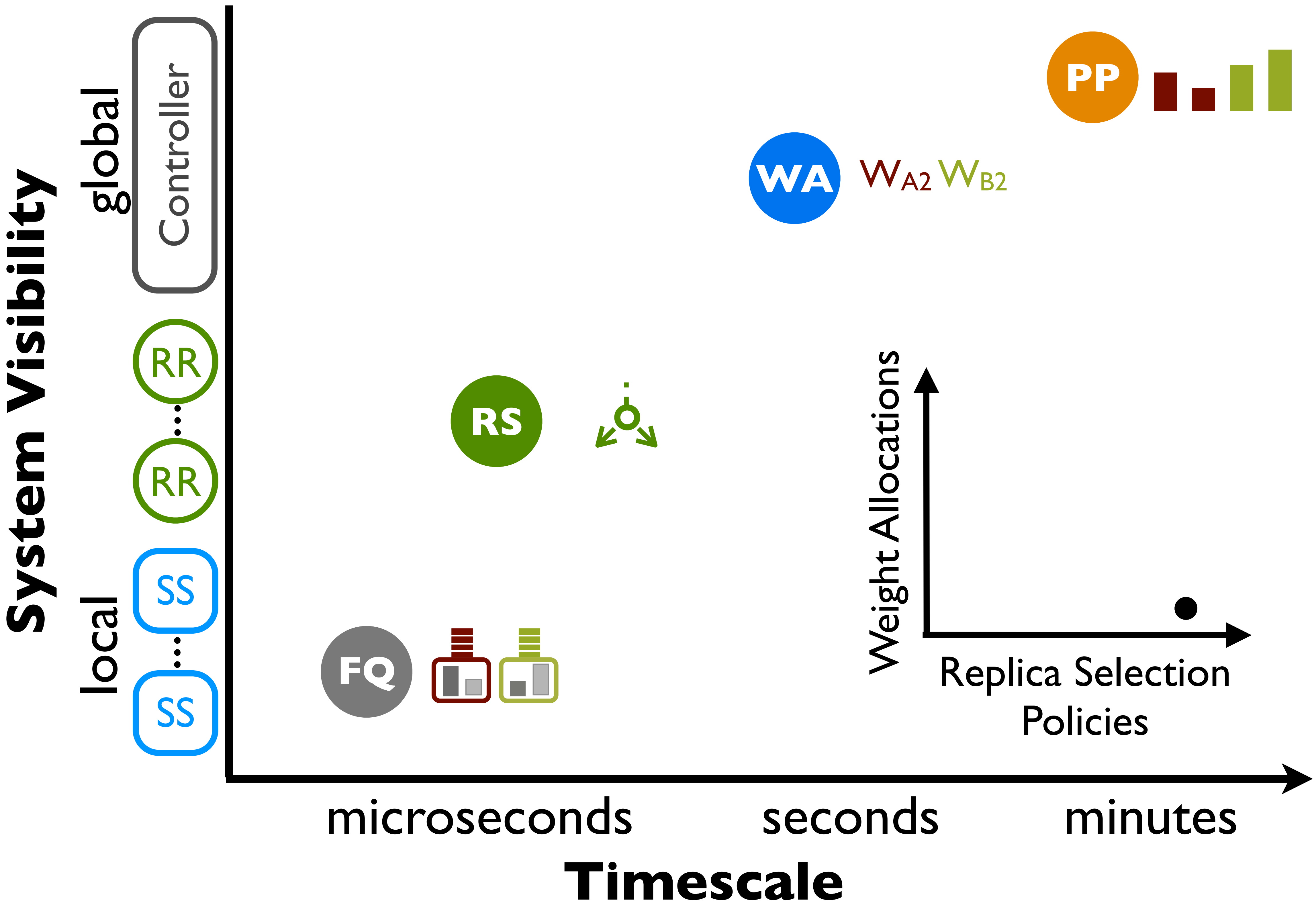
Pisces Mechanisms Solve For Global Fairness



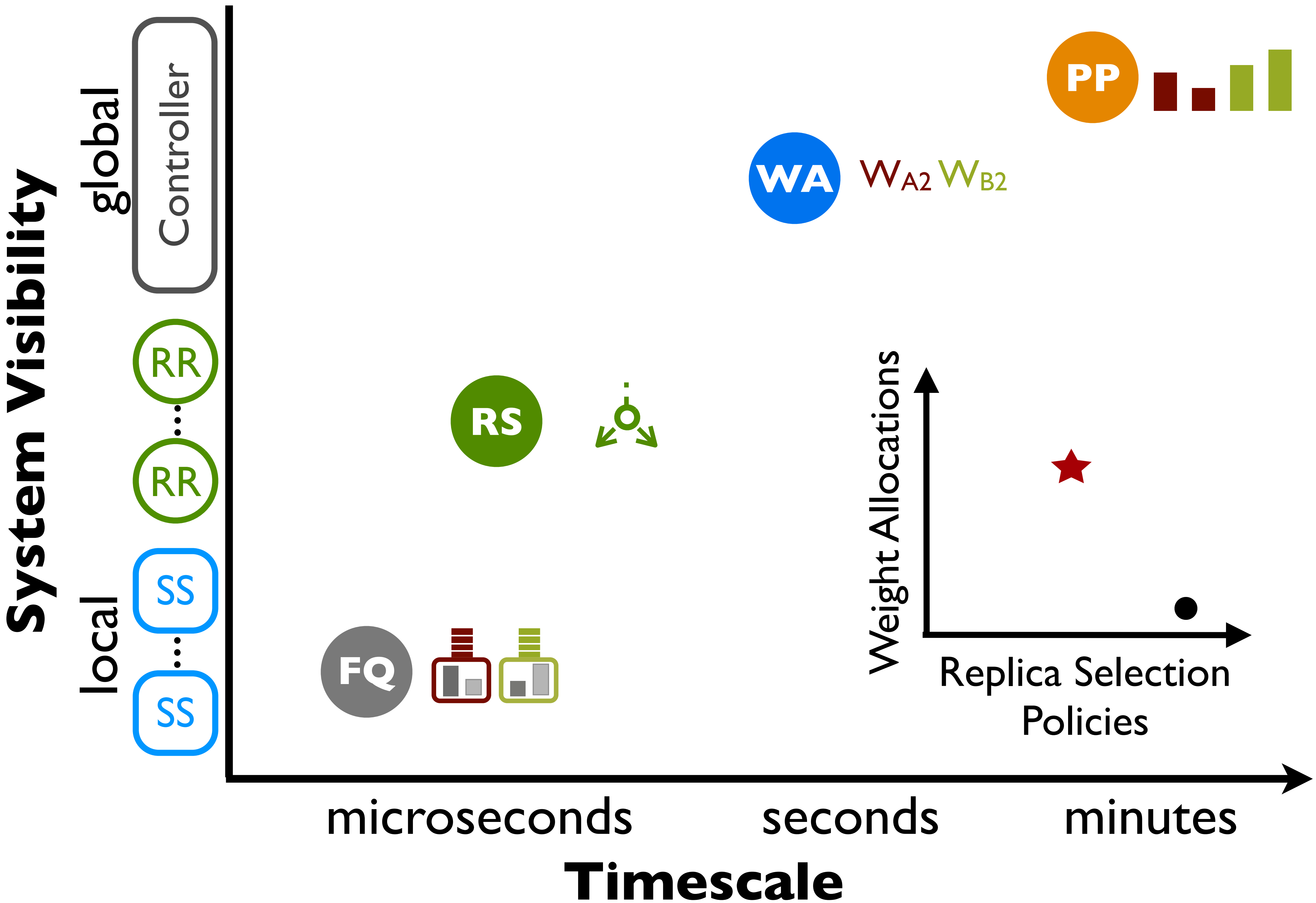
Pisces Mechanisms Solve For Global Fairness



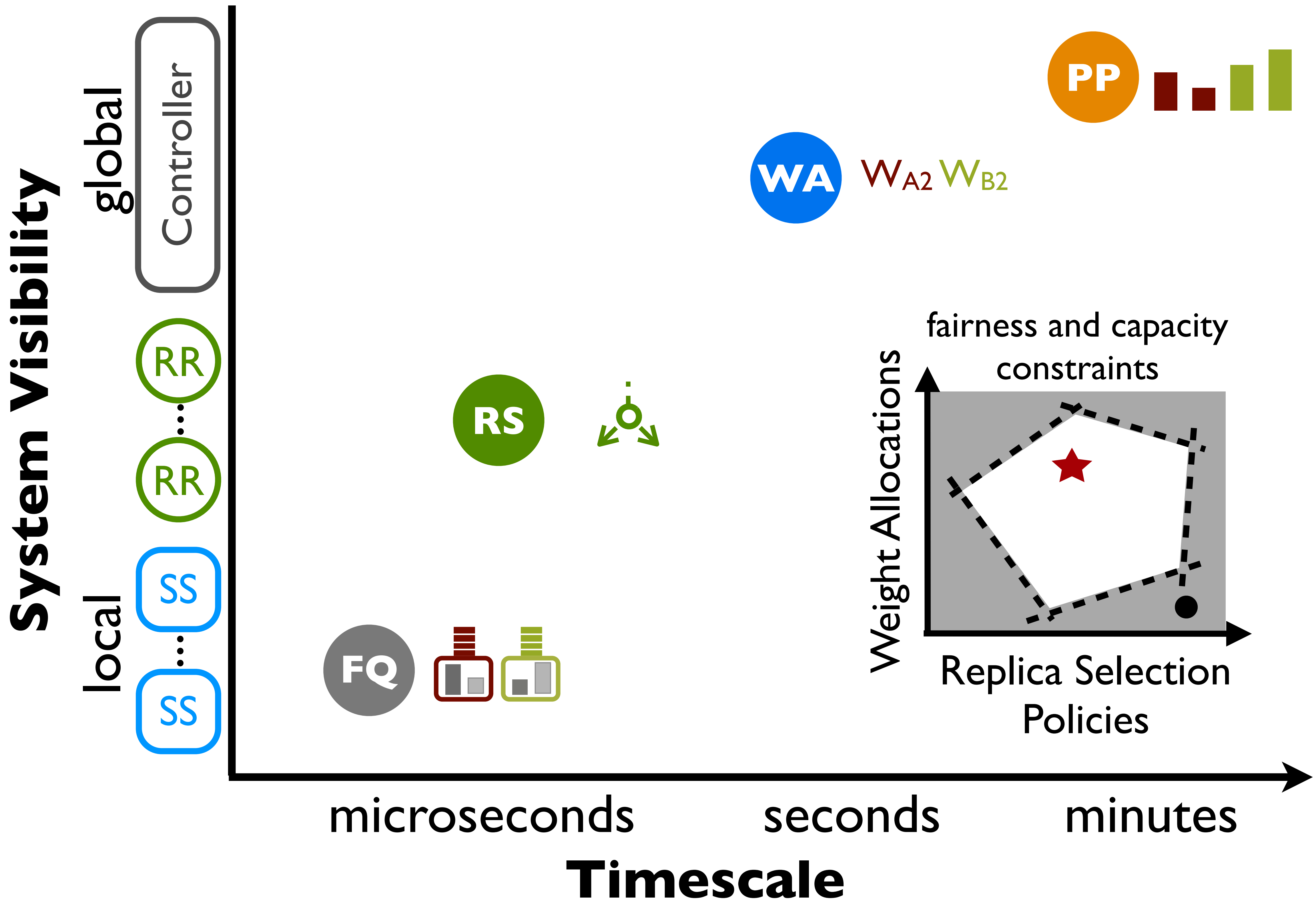
Pisces Mechanisms Solve For Global Fairness



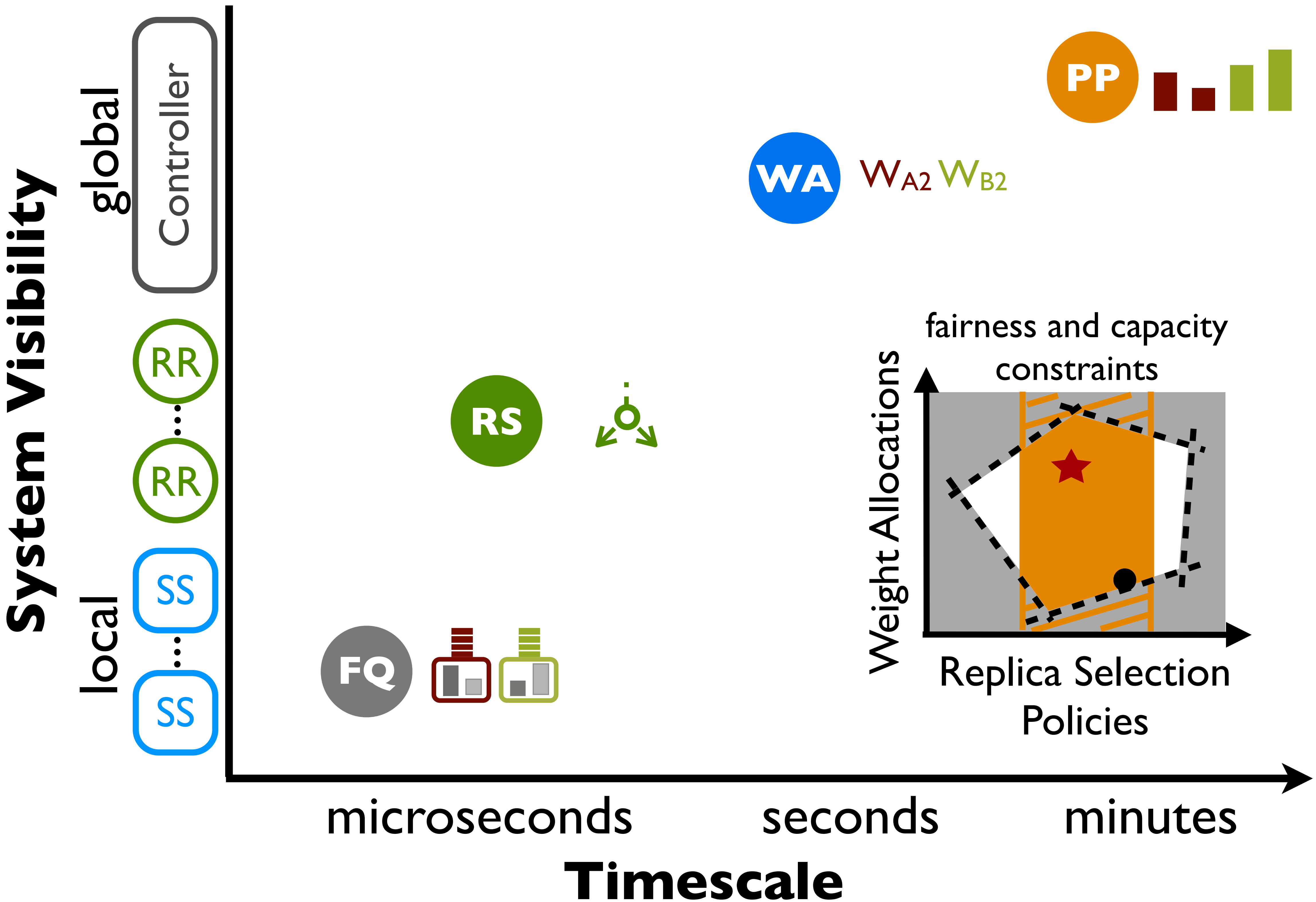
Pisces Mechanisms Solve For Global Fairness



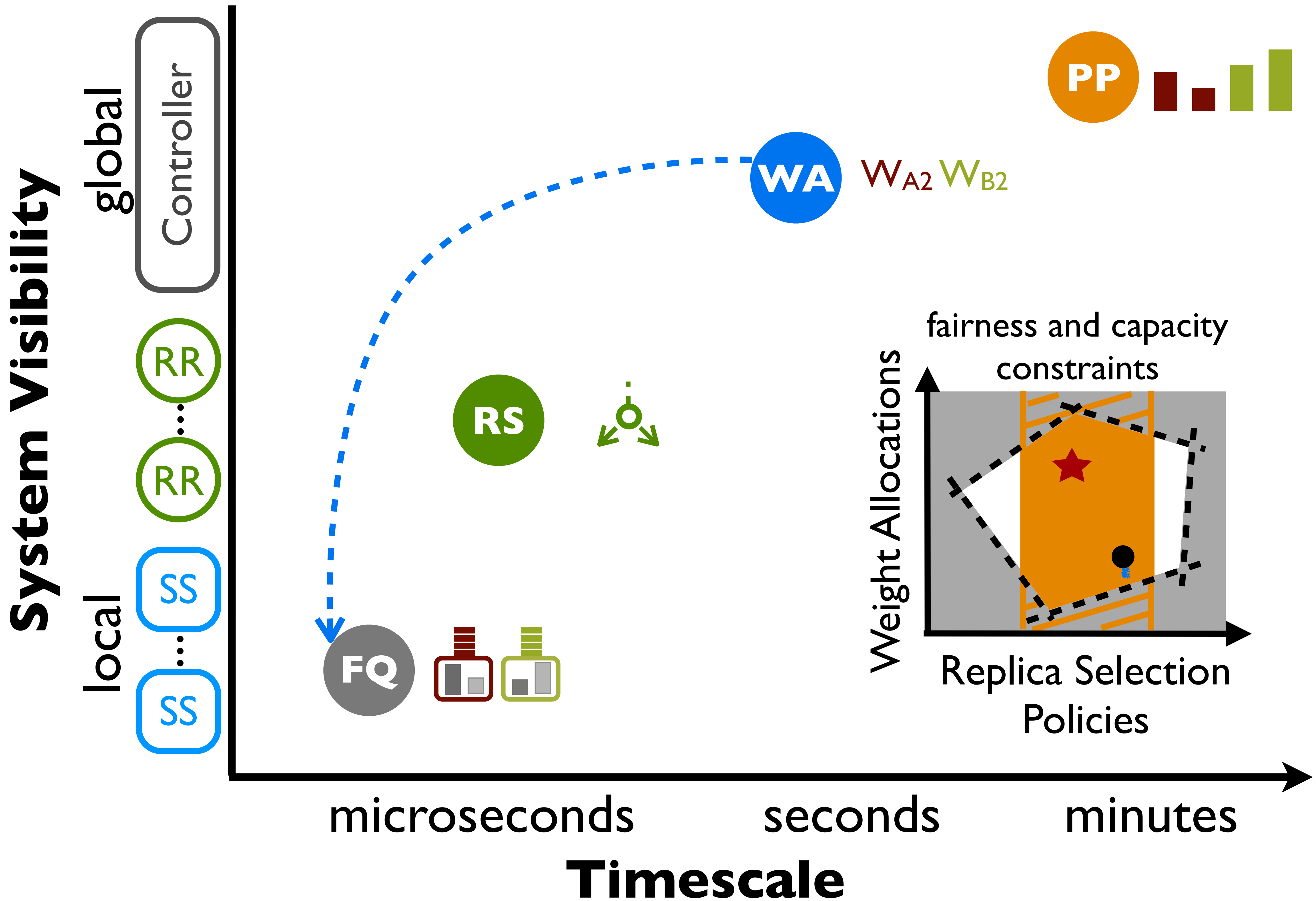
Pisces Mechanisms Solve For Global Fairness



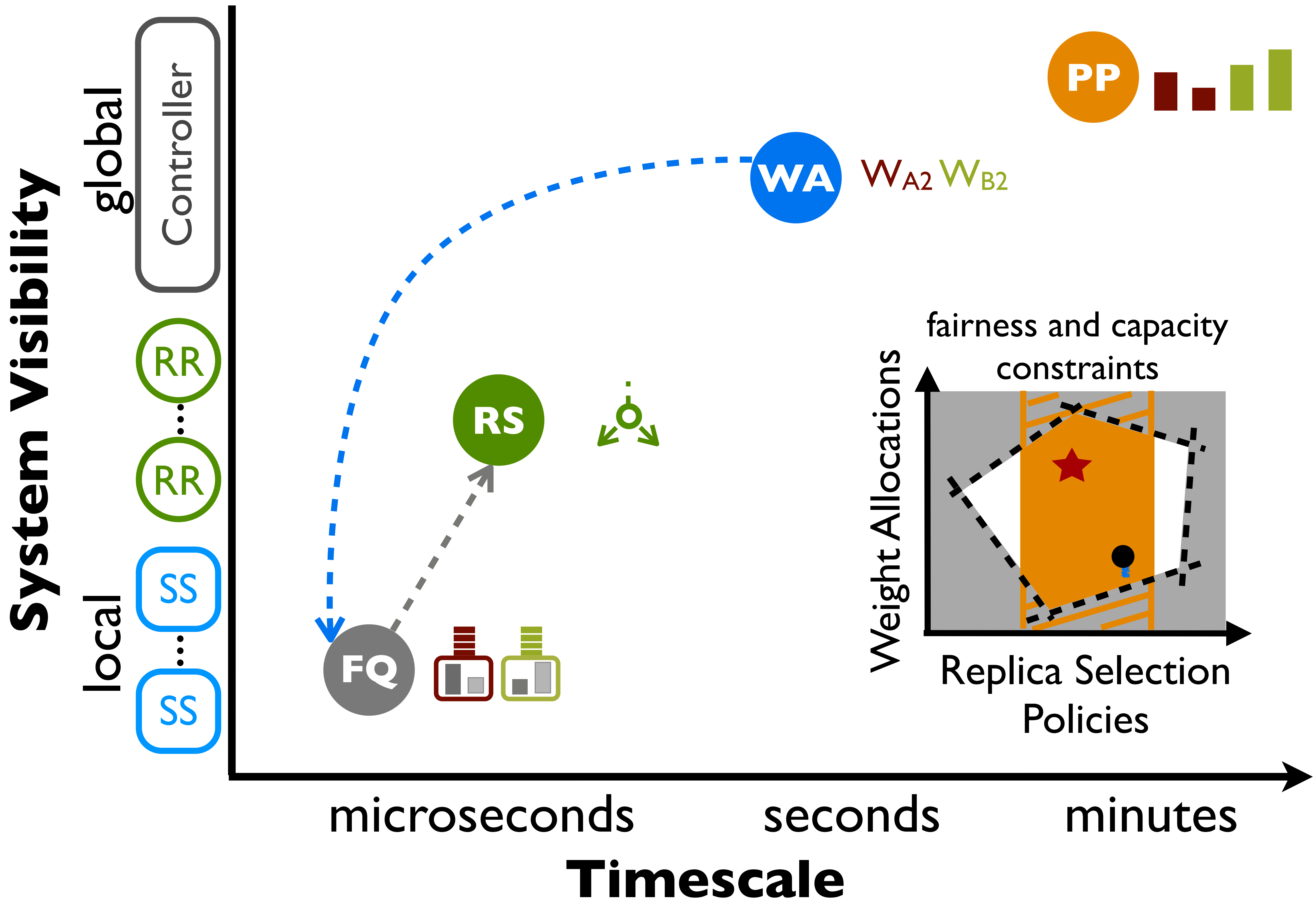
Pisces Mechanisms Solve For Global Fairness



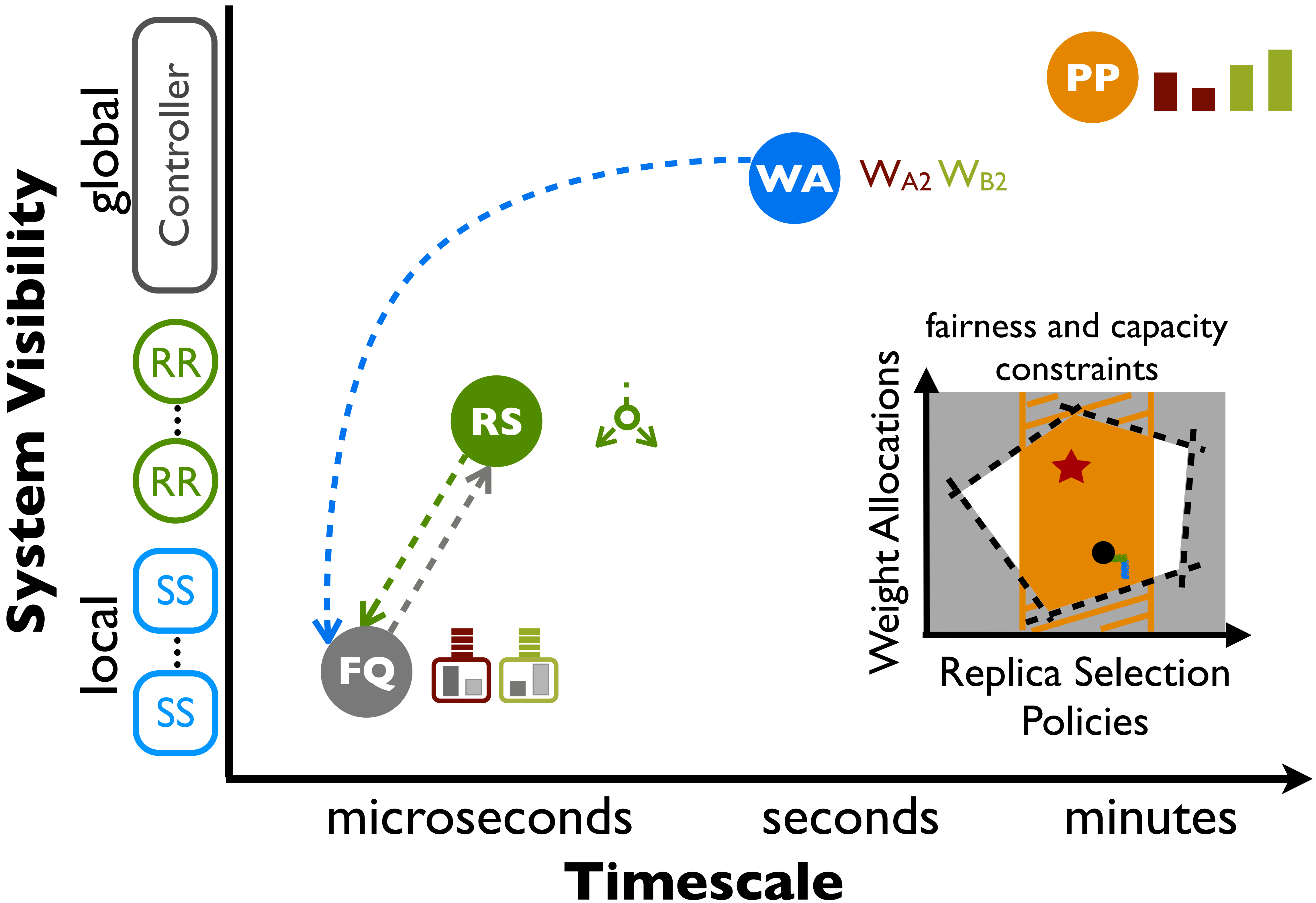
Pisces Mechanisms Solve For Global Fairness



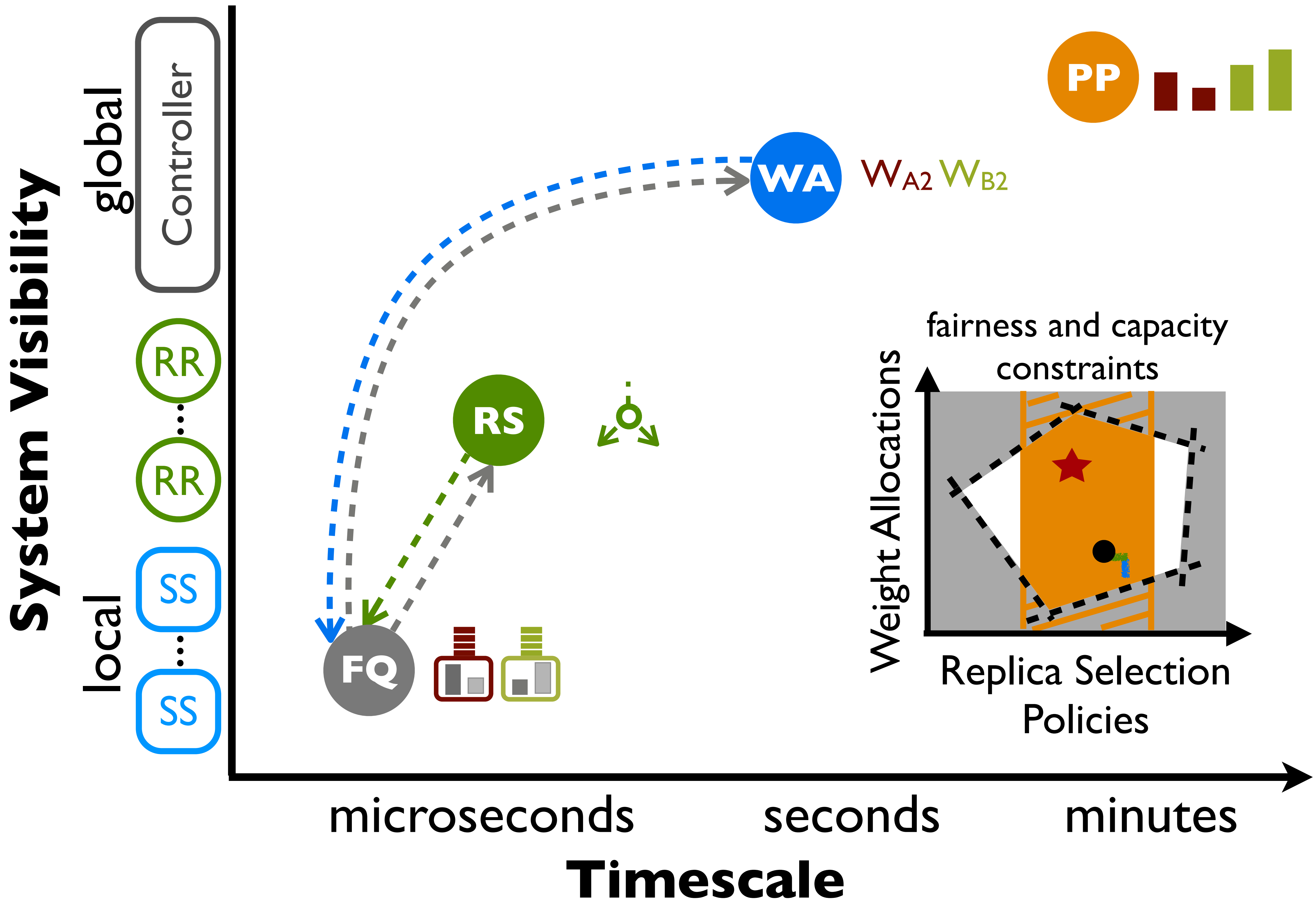
Pisces Mechanisms Solve For Global Fairness



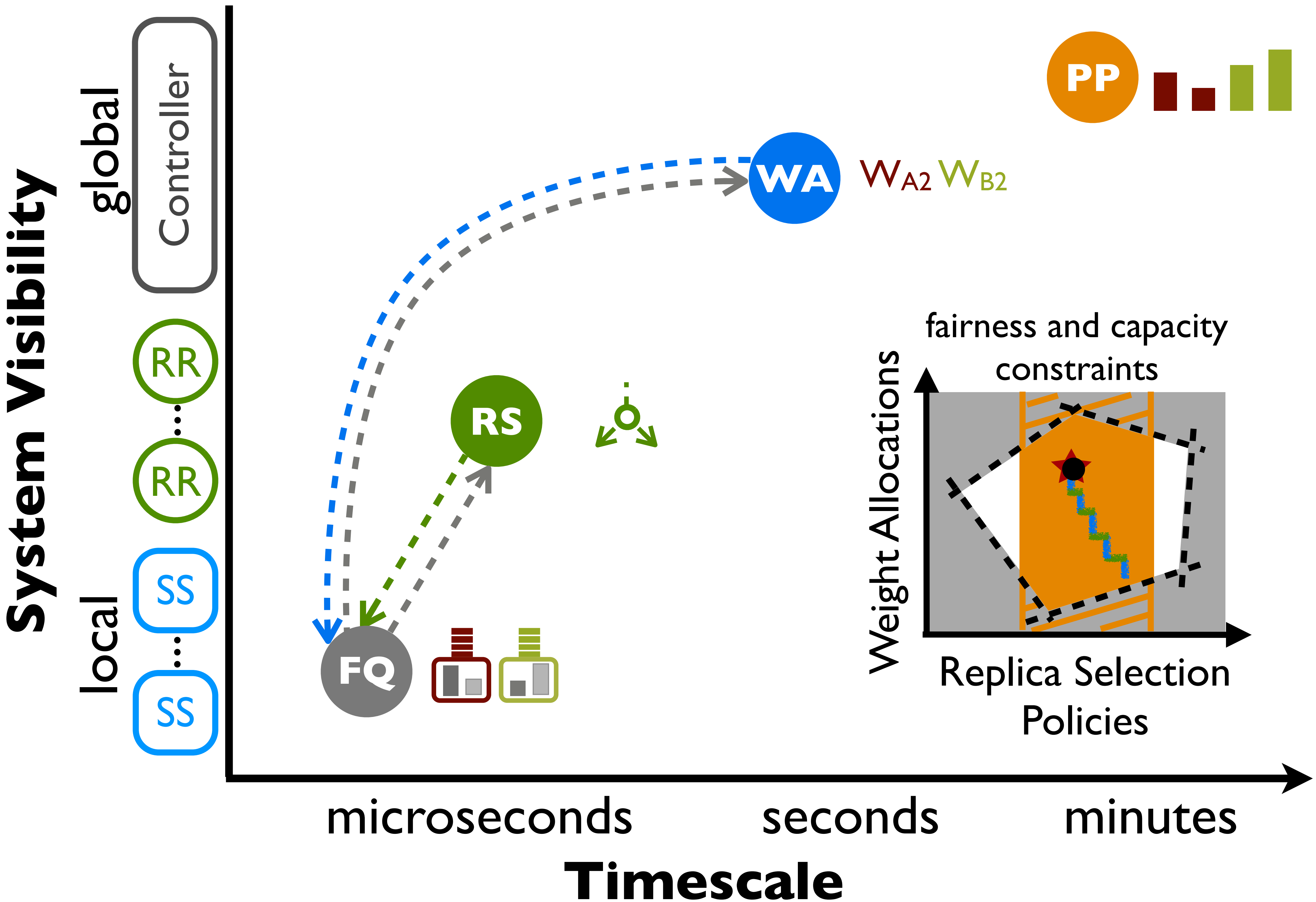
Pisces Mechanisms Solve For Global Fairness



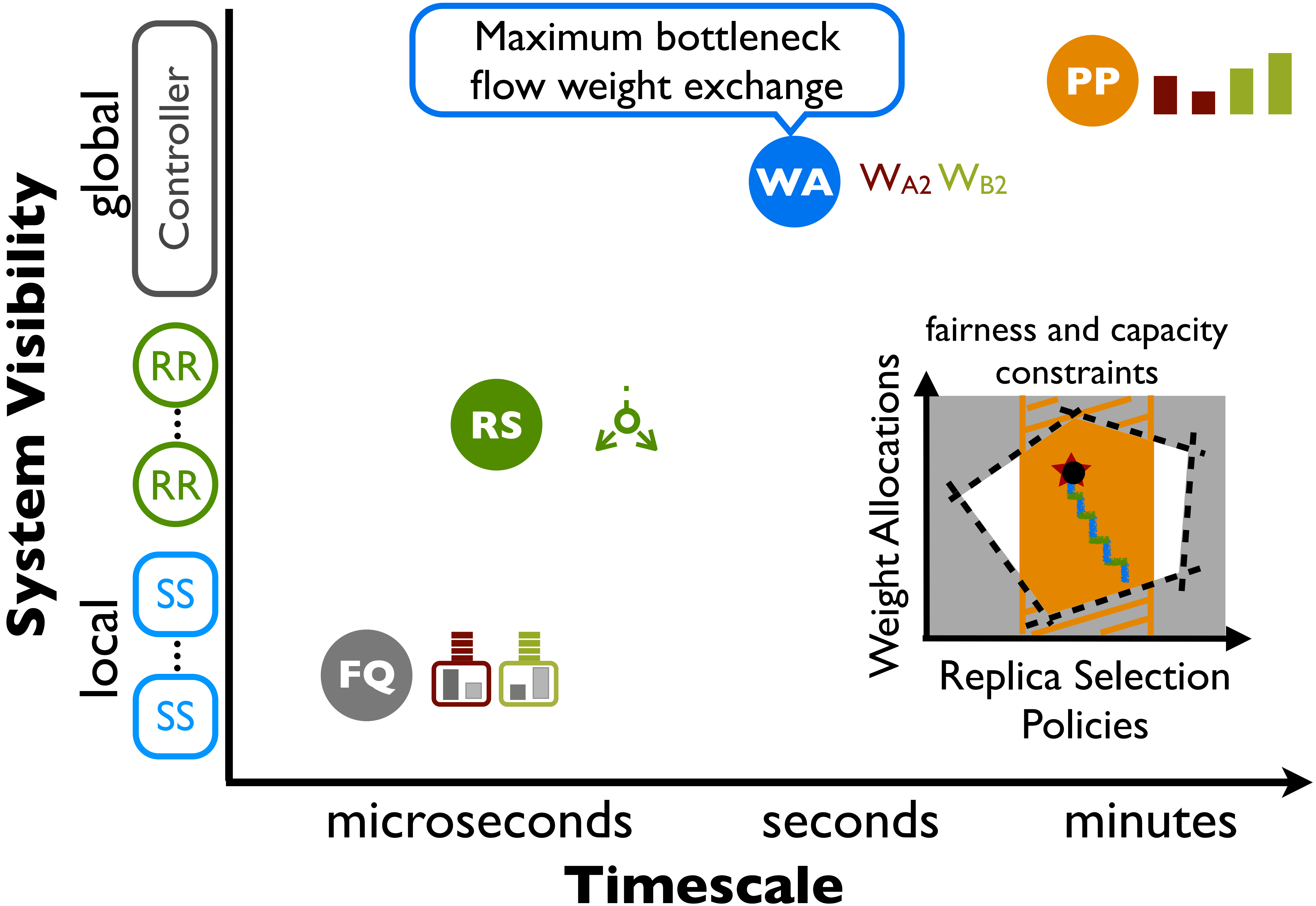
Pisces Mechanisms Solve For Global Fairness



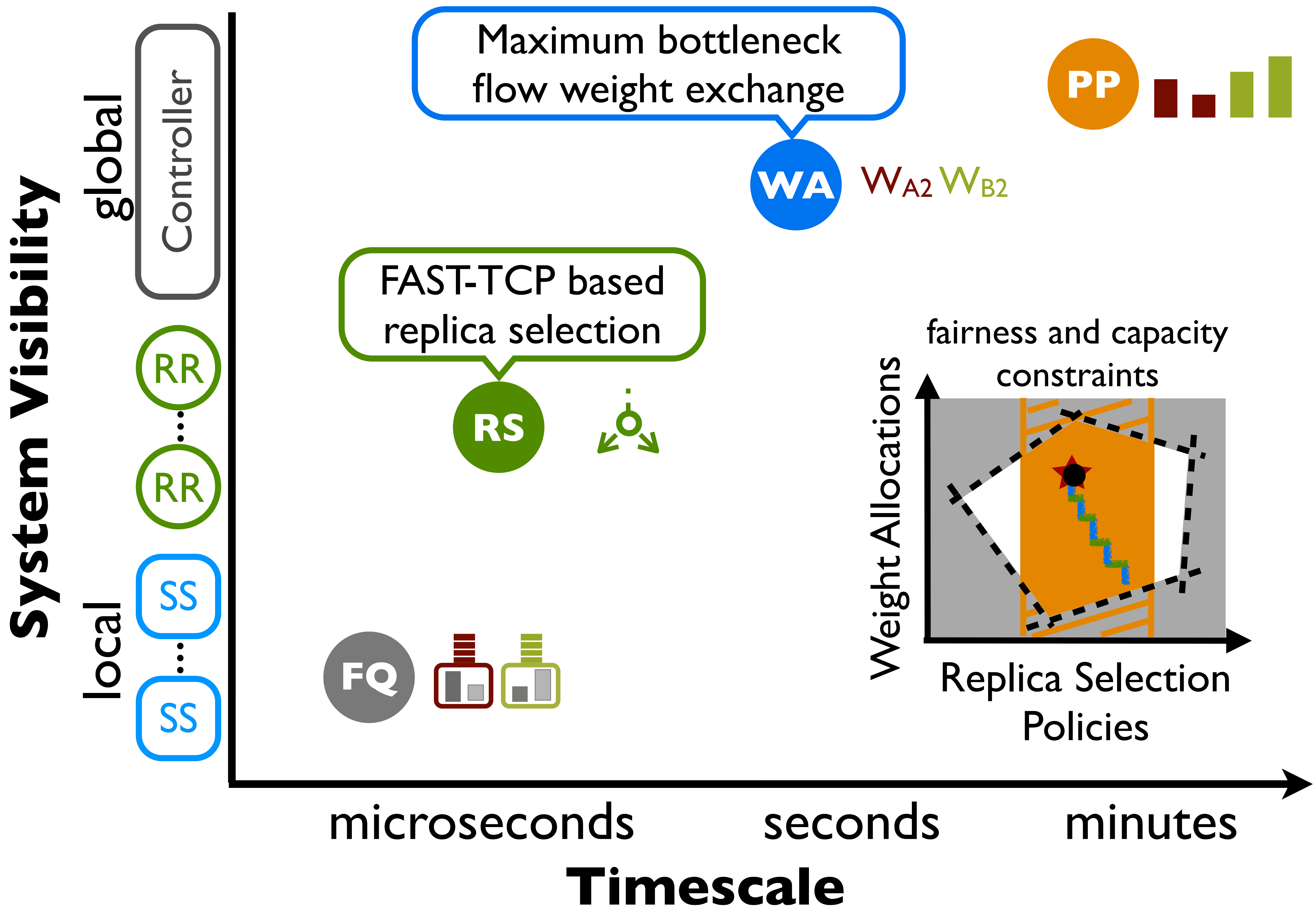
Pisces Mechanisms Solve For Global Fairness



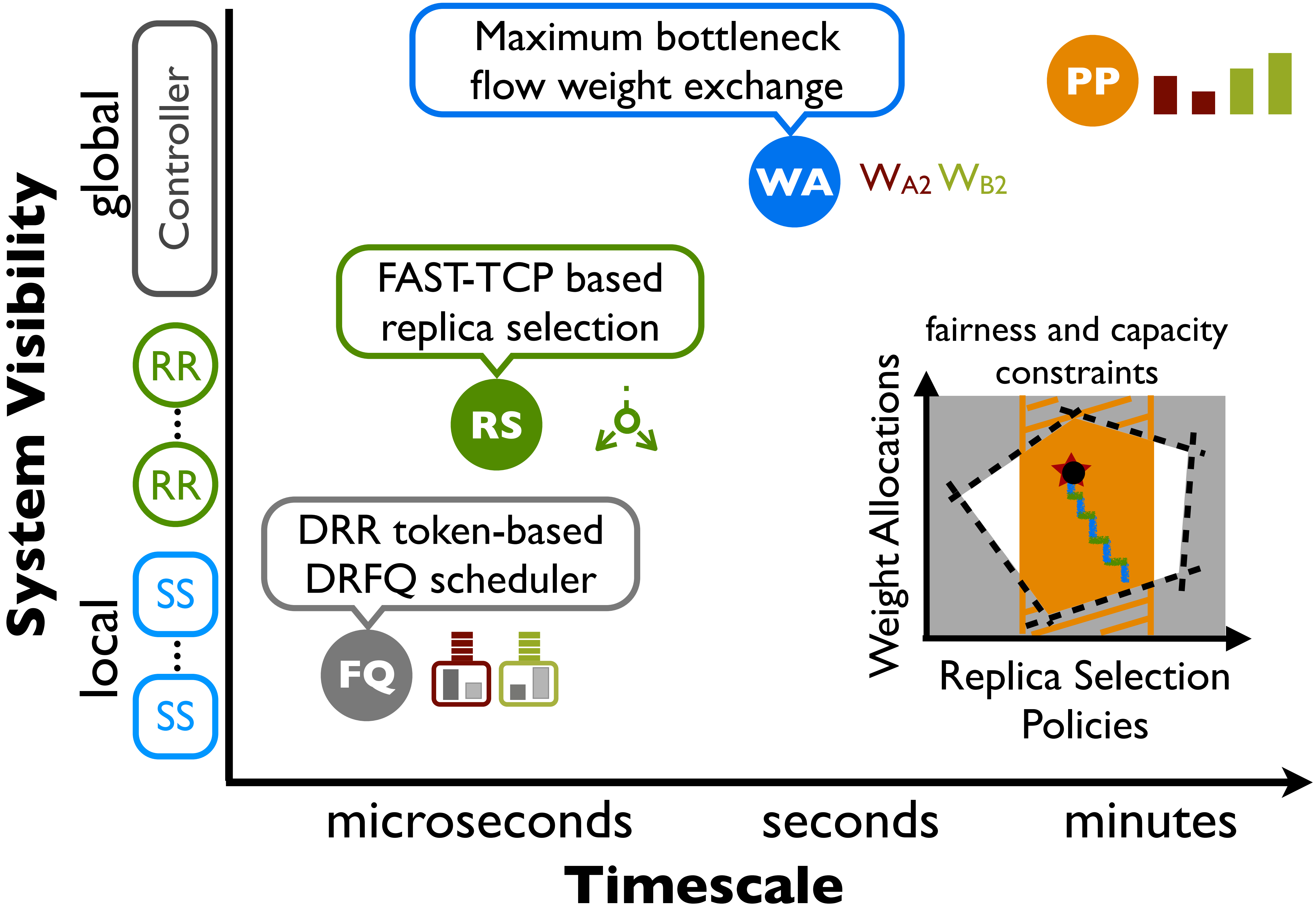
Pisces Mechanisms Solve For Global Fairness



Pisces Mechanisms Solve For Global Fairness



Pisces Mechanisms Solve For Global Fairness



Evaluation

Evaluation

- Does Pisces achieve (even) system-wide fairness?

Evaluation

- Does Pisces achieve (even) system-wide fairness?
  - Is each Pisces mechanism necessary for fairness?

Evaluation

- Does Pisces achieve (even) system-wide fairness?
  - Is each Pisces mechanism necessary for fairness?
  - What is the overhead of using Pisces?

Evaluation

- Does Pisces achieve (even) system-wide fairness?
  - Is each Pisces mechanism necessary for fairness?
  - What is the overhead of using Pisces?
- Does Pisces handle mixed workloads?

Evaluation

- Does Pisces achieve (even) system-wide fairness?
  - Is each Pisces mechanism necessary for fairness?
  - What is the overhead of using Pisces?
- Does Pisces handle mixed workloads?
- Does Pisces provide weighted system-wide fairness?

Evaluation

- Does Pisces achieve (even) system-wide fairness?
  - Is each Pisces mechanism necessary for fairness?
  - What is the overhead of using Pisces?
- Does Pisces handle mixed workloads?
- Does Pisces provide weighted system-wide fairness?
- Does Pisces provide local dominant resource fairness?

Evaluation

- Does Pisces achieve (even) system-wide fairness?
  - Is each Pisces mechanism necessary for fairness?
  - What is the overhead of using Pisces?
- Does Pisces handle mixed workloads?
- Does Pisces provide weighted system-wide fairness?
- Does Pisces provide local dominant resource fairness?
- Does Pisces handle dynamic demand?

Evaluation

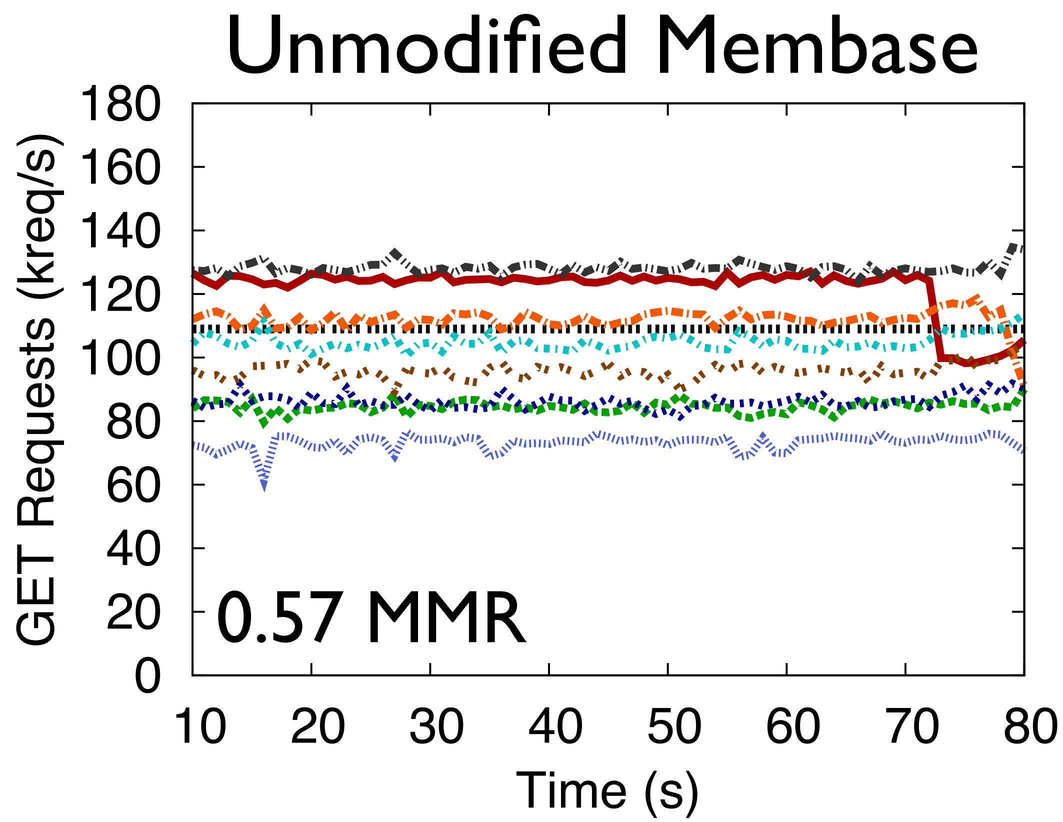
- Does Pisces achieve (even) system-wide fairness?
  - Is each Pisces mechanism necessary for fairness?
  - What is the overhead of using Pisces?
- Does Pisces handle mixed workloads?
- Does Pisces provide weighted system-wide fairness?
- Does Pisces provide local dominant resource fairness?
- Does Pisces handle dynamic demand?
- Does Pisces adapt to changes in object popularity?

Evaluation

- Does Pisces achieve (even) system-wide fairness?
  - Is each Pisces mechanism necessary for fairness?
  - What is the overhead of using Pisces?
- Does Pisces handle mixed workloads?
- Does Pisces provide weighted system-wide fairness?
- Does Pisces provide local dominant resource fairness?
- Does Pisces handle dynamic demand?
- Does Pisces adapt to changes in object popularity?

Pisces Achieves System-wide Per-tenant Fairness

Ideal fair share: 110 kreq/s (1kB requests)



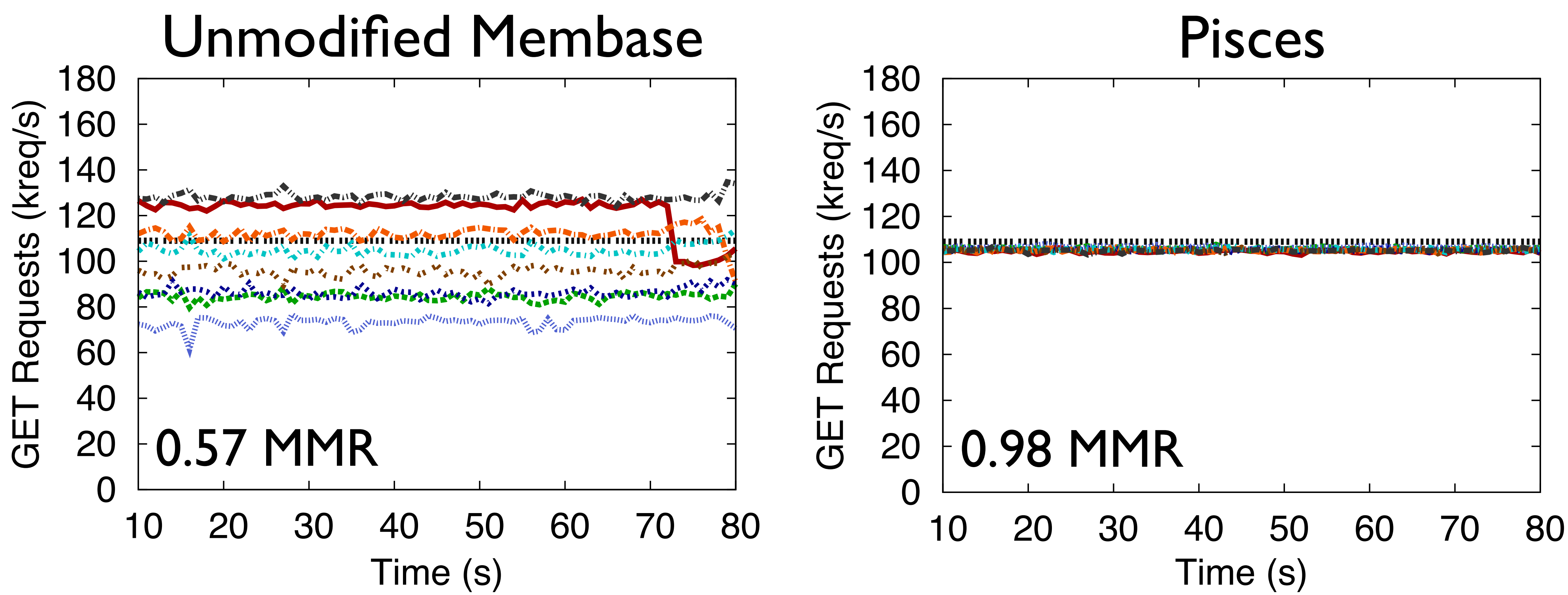
8 Tenants - 8 Client - 8 Storage Nodes

Zipfian object popularity distribution

Min-Max Ratio: min rate/max rate (0,1]

Pisces Achieves System-wide Per-tenant Fairness

Ideal fair share: 110 kreq/s (1kB requests)



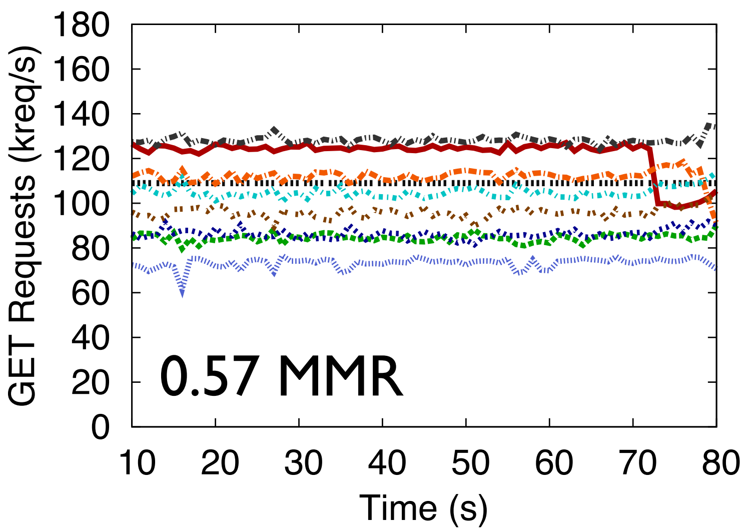
8 Tenants - 8 Client - 8 Storage Nodes

Zipfian object popularity distribution

Min-Max Ratio: min rate/max rate (0,1]

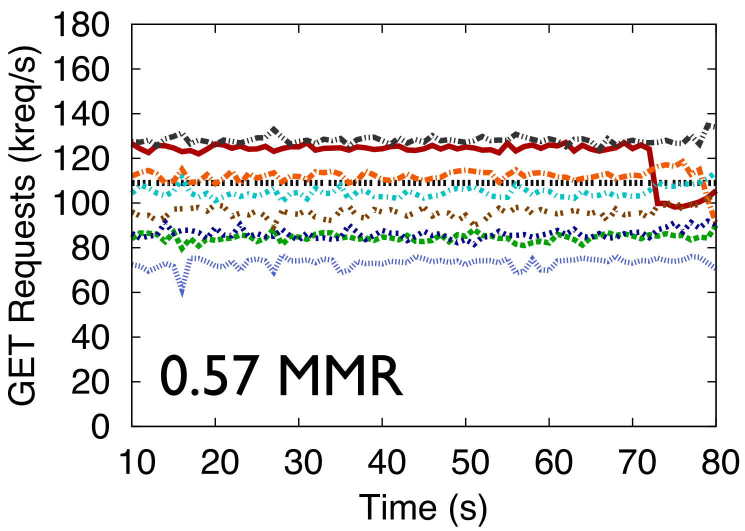
Each Pisces Mechanism Contributes to System-wide Fairness and Isolation

Unmodified Membase

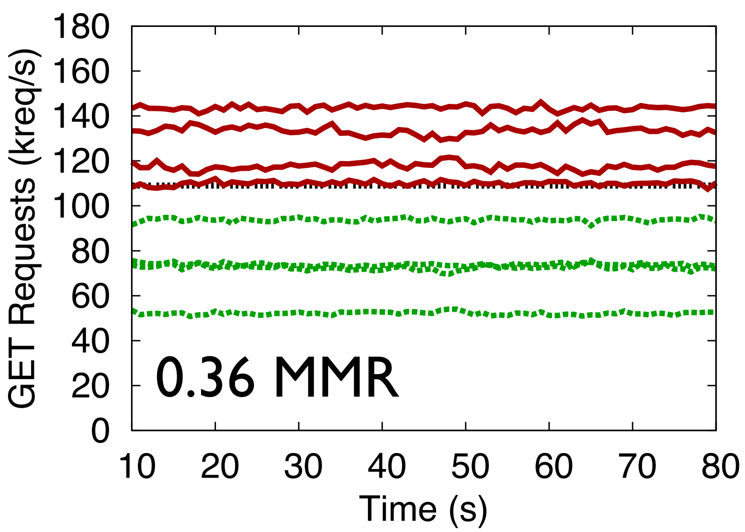


Each Pisces Mechanism Contributes to System-wide Fairness and Isolation

Unmodified Membase



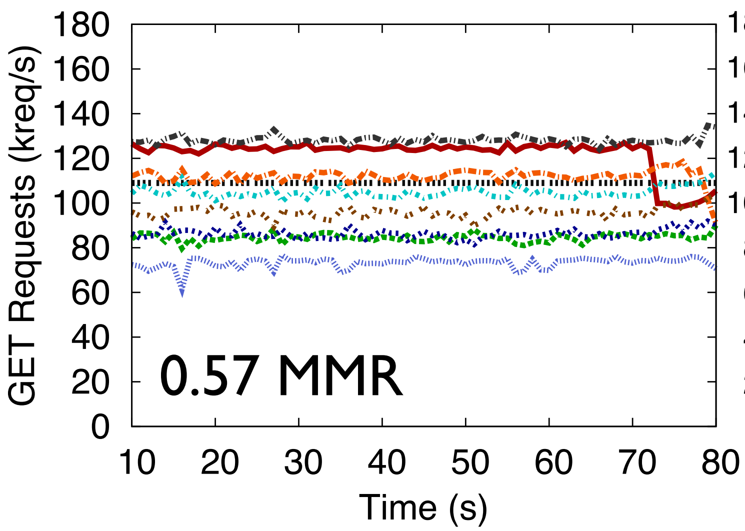
2x vs 1x demand

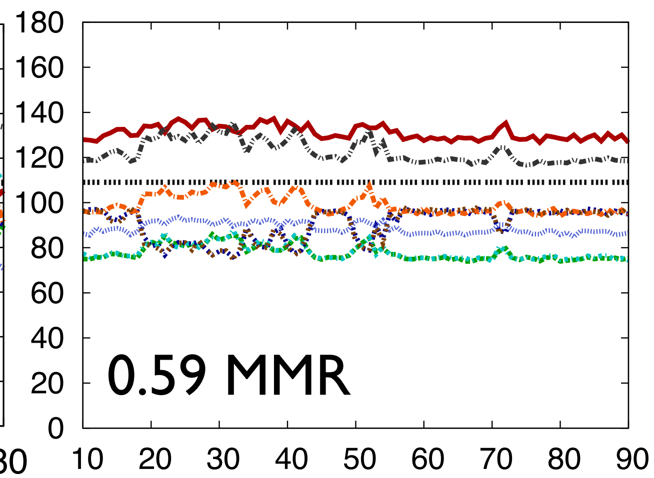


Each Pisces Mechanism Contributes to System-wide Fairness and Isolation

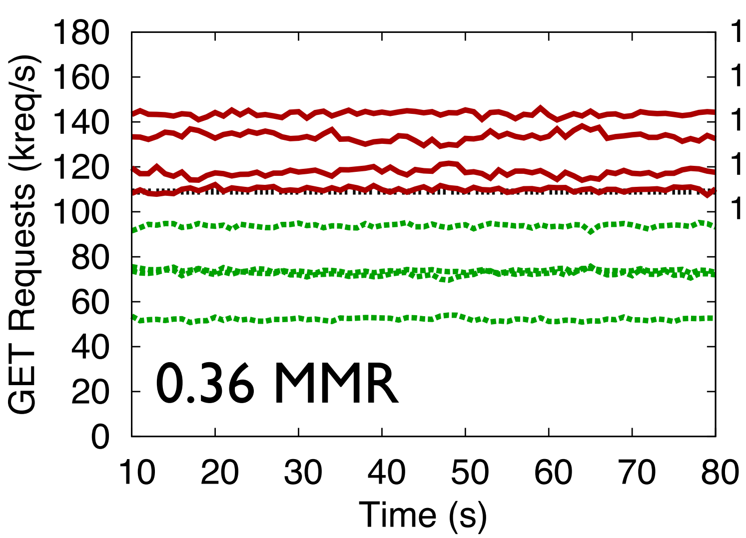
Unmodified Membase

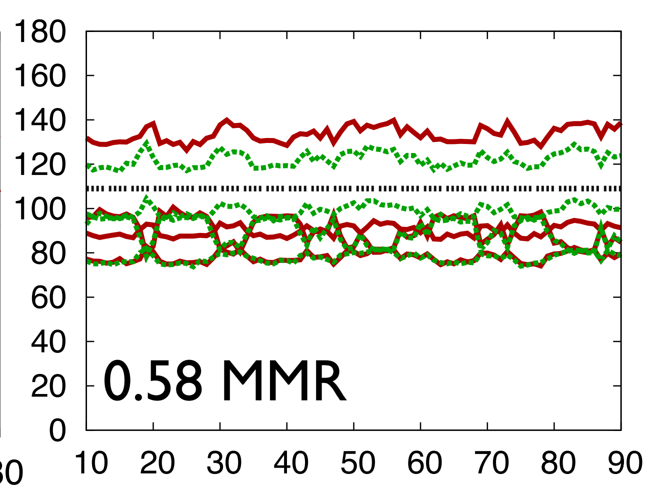
FQ





2x vs 1x demand



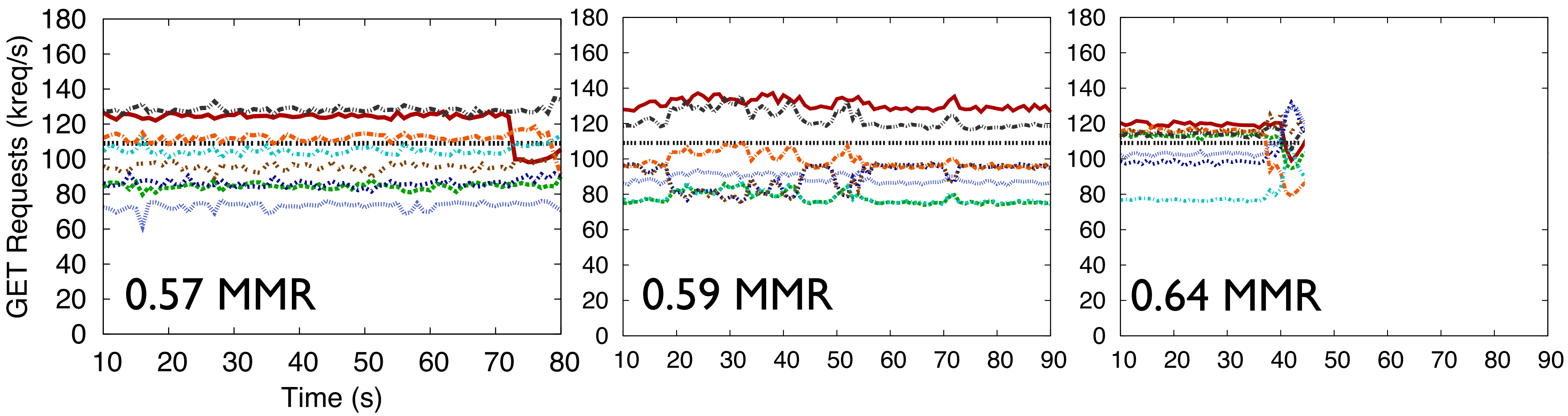


Each Pisces Mechanism Contributes to System-wide Fairness and Isolation

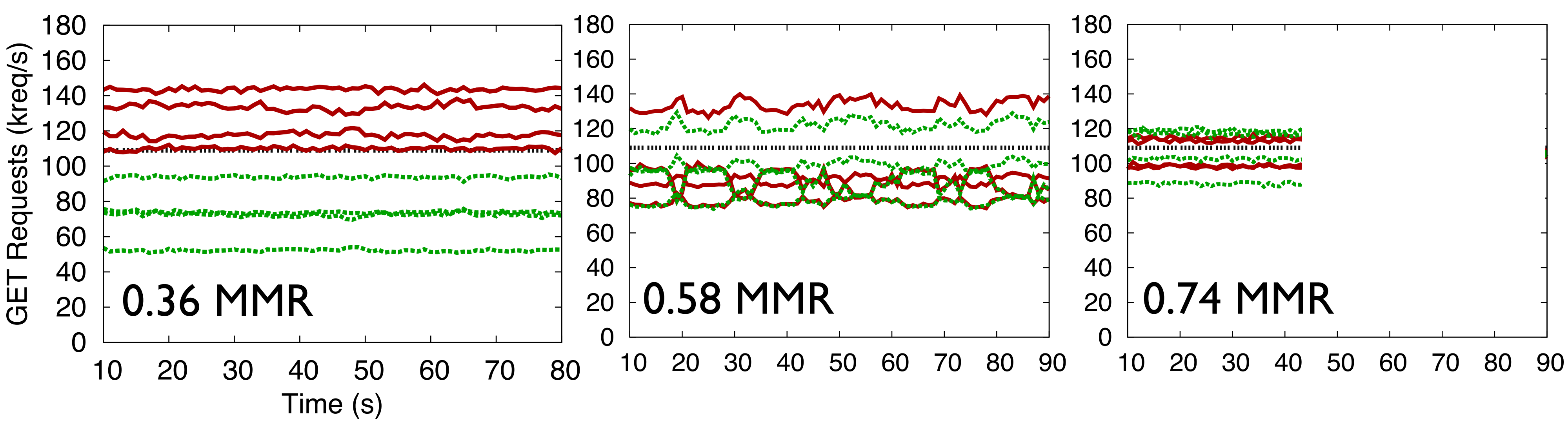
Unmodified Membase

FQ

FQ PP



2x vs 1x demand

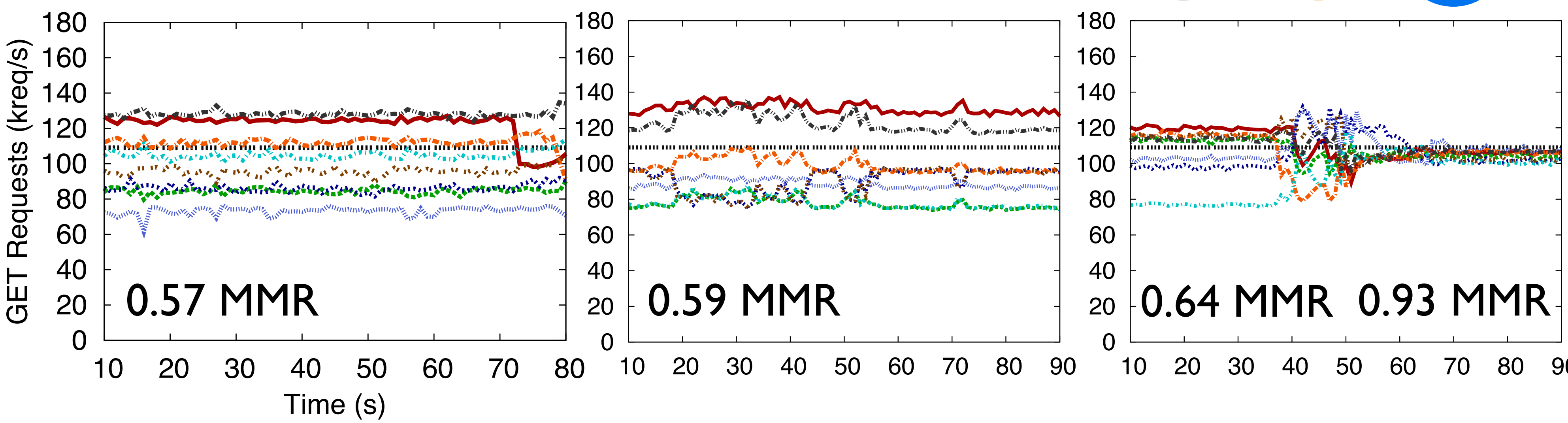


Each Pisces Mechanism Contributes to System-wide Fairness and Isolation

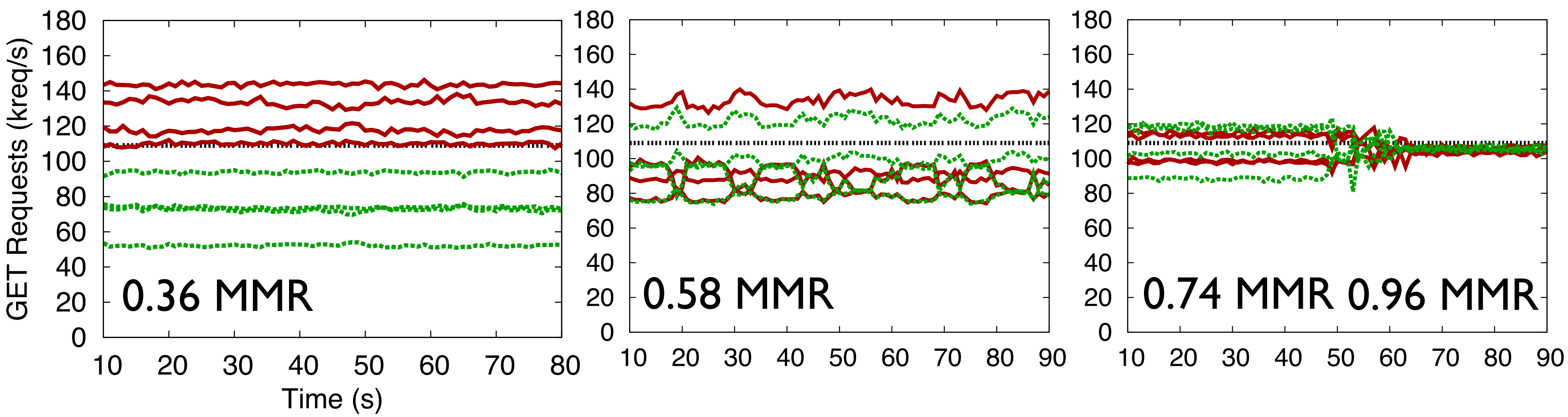
Unmodified Membase

FQ

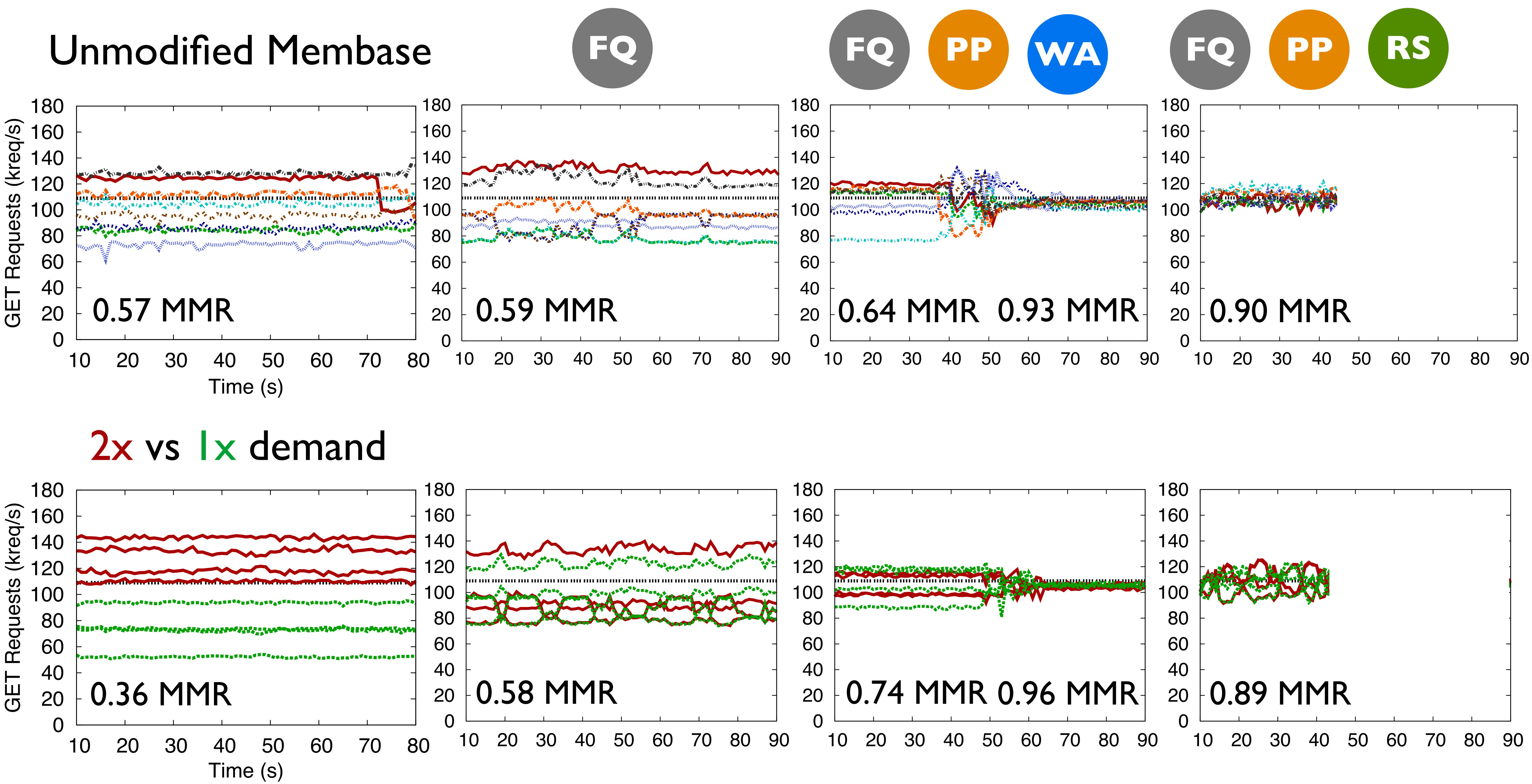
FQ PP WA



2x vs 1x demand

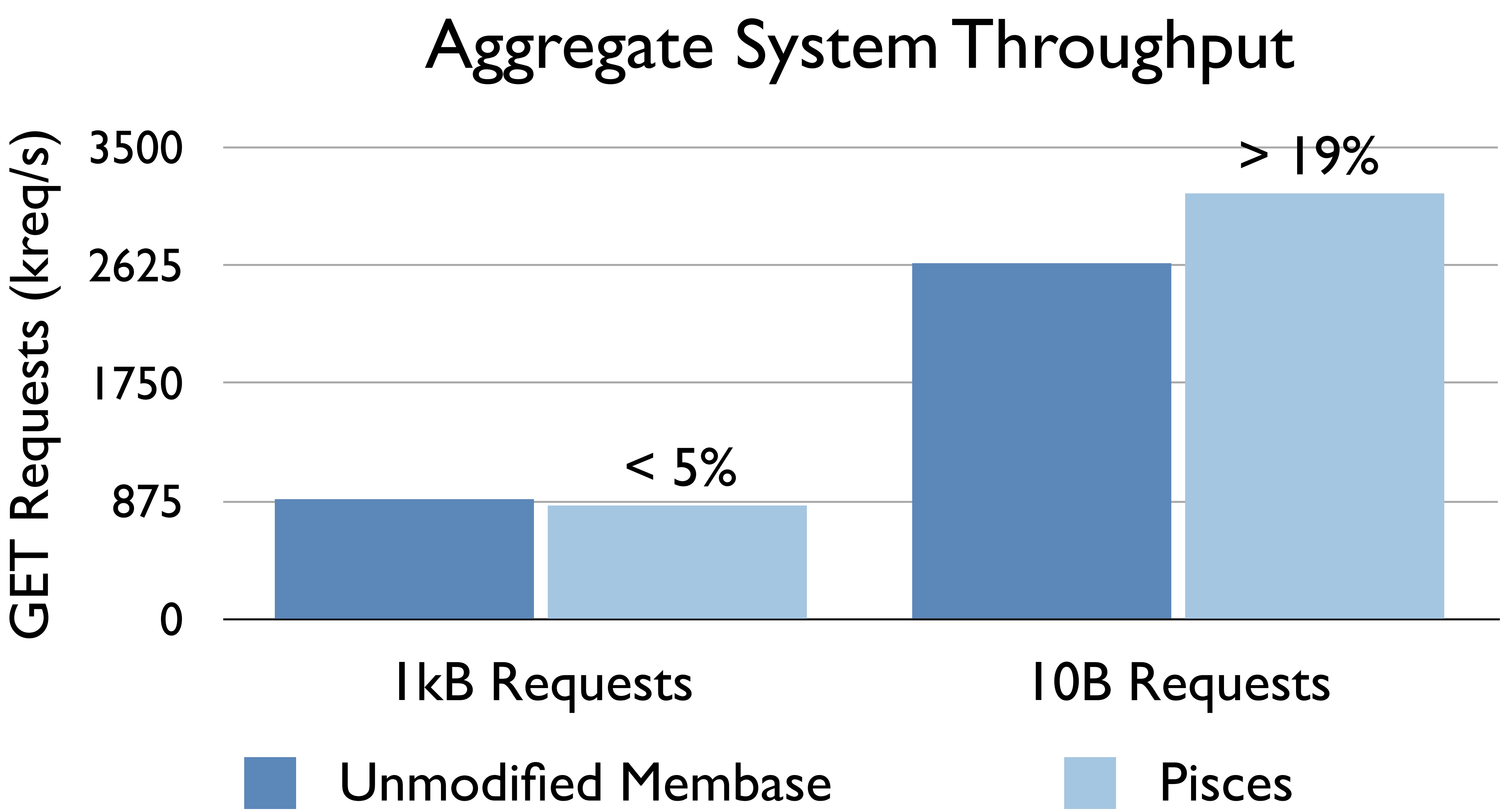


Each Pisces Mechanism Contributes to System-wide Fairness and Isolation



Each Pisces Mechanism Contributes to System-wide Fairness and Isolation

Pisces Imposes Low-overhead

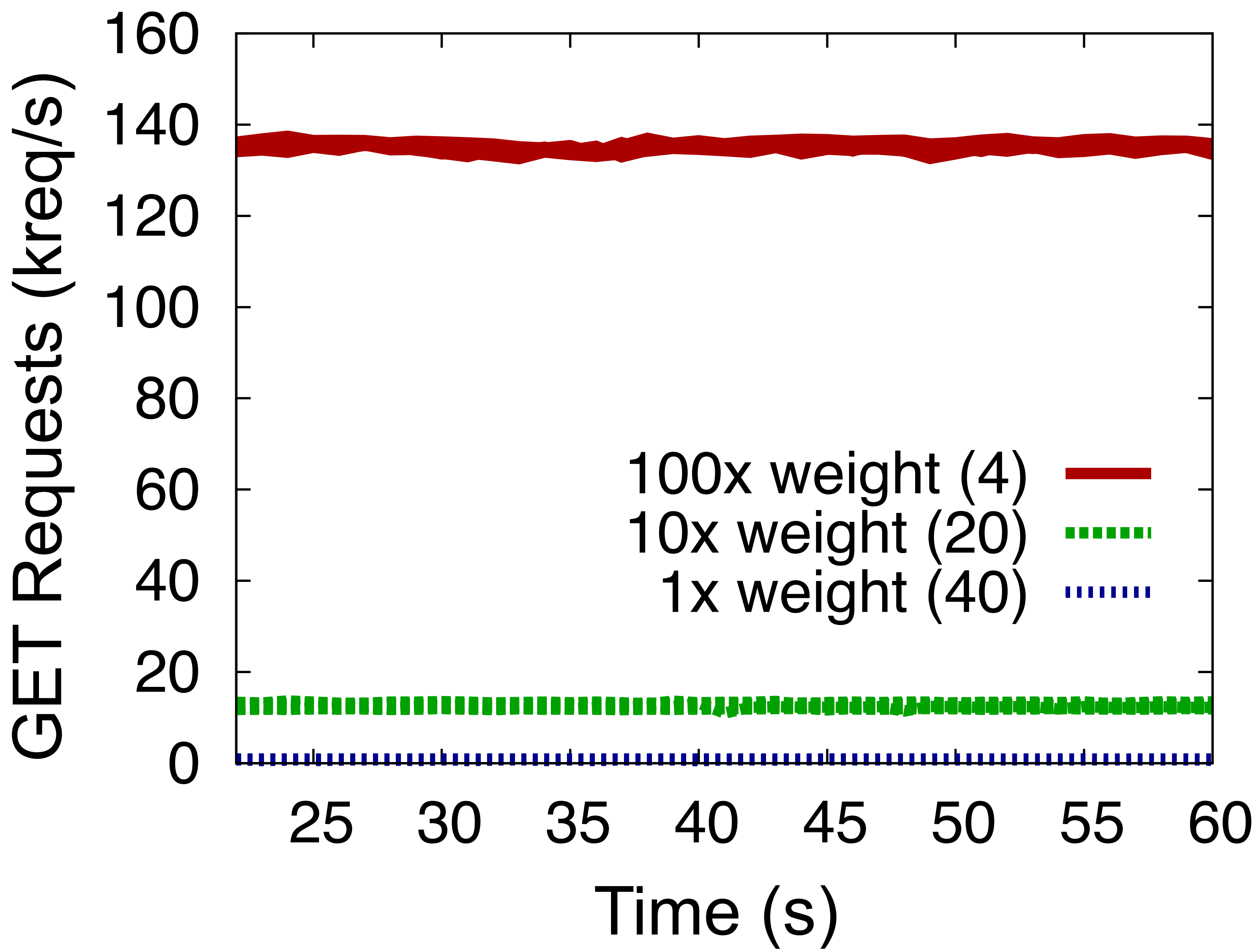


Pisces Achieves System-wide Weighted Fairness

4 heavy hitters   20 moderate demand   40 low demand

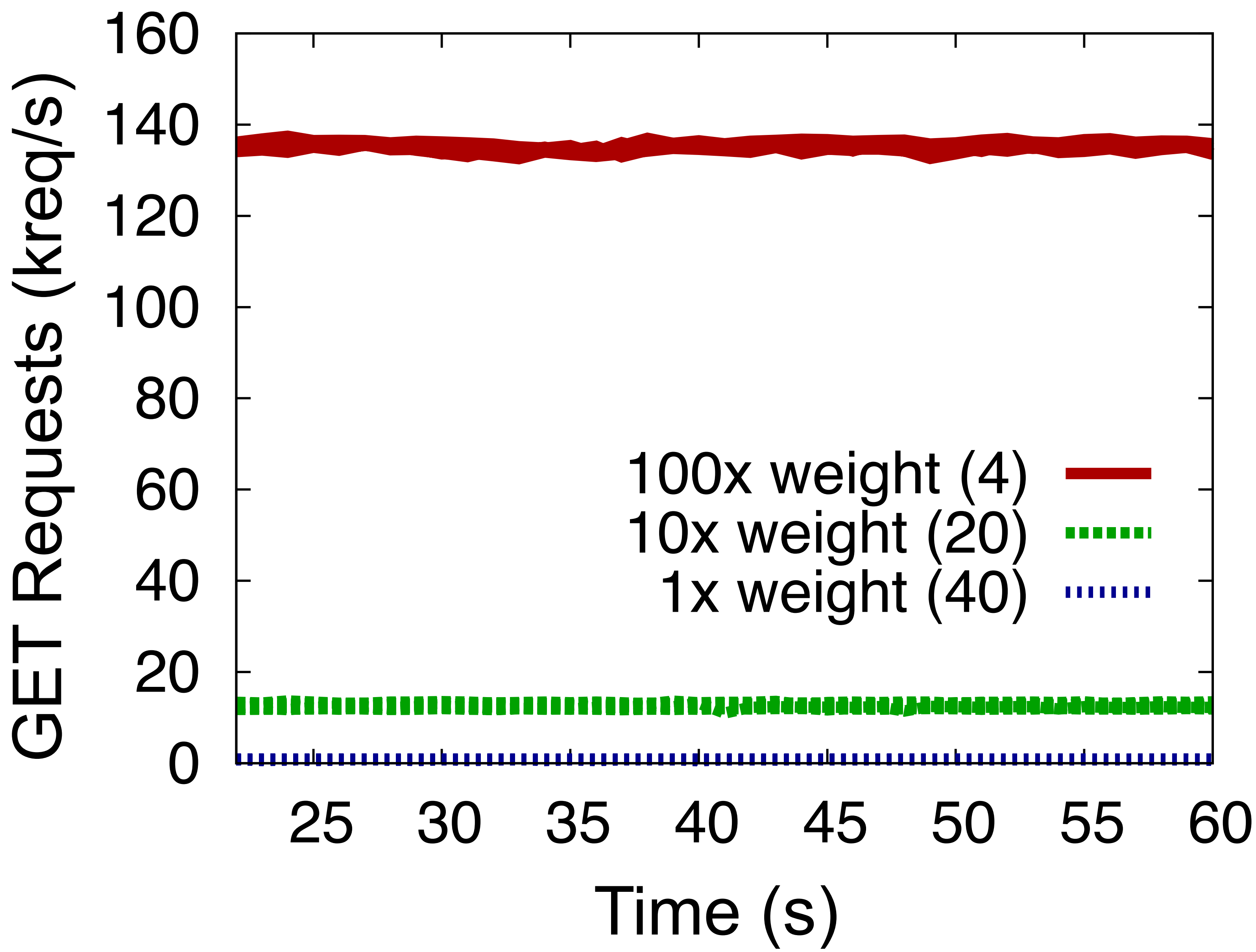
Pisces Achieves System-wide Weighted Fairness

4 heavy hitters   20 moderate demand   40 low demand



Pisces Achieves System-wide Weighted Fairness

0.98 MMR	0.89 MMR	0.91 MMR
4 heavy hitters	20 moderate demand	40 low demand

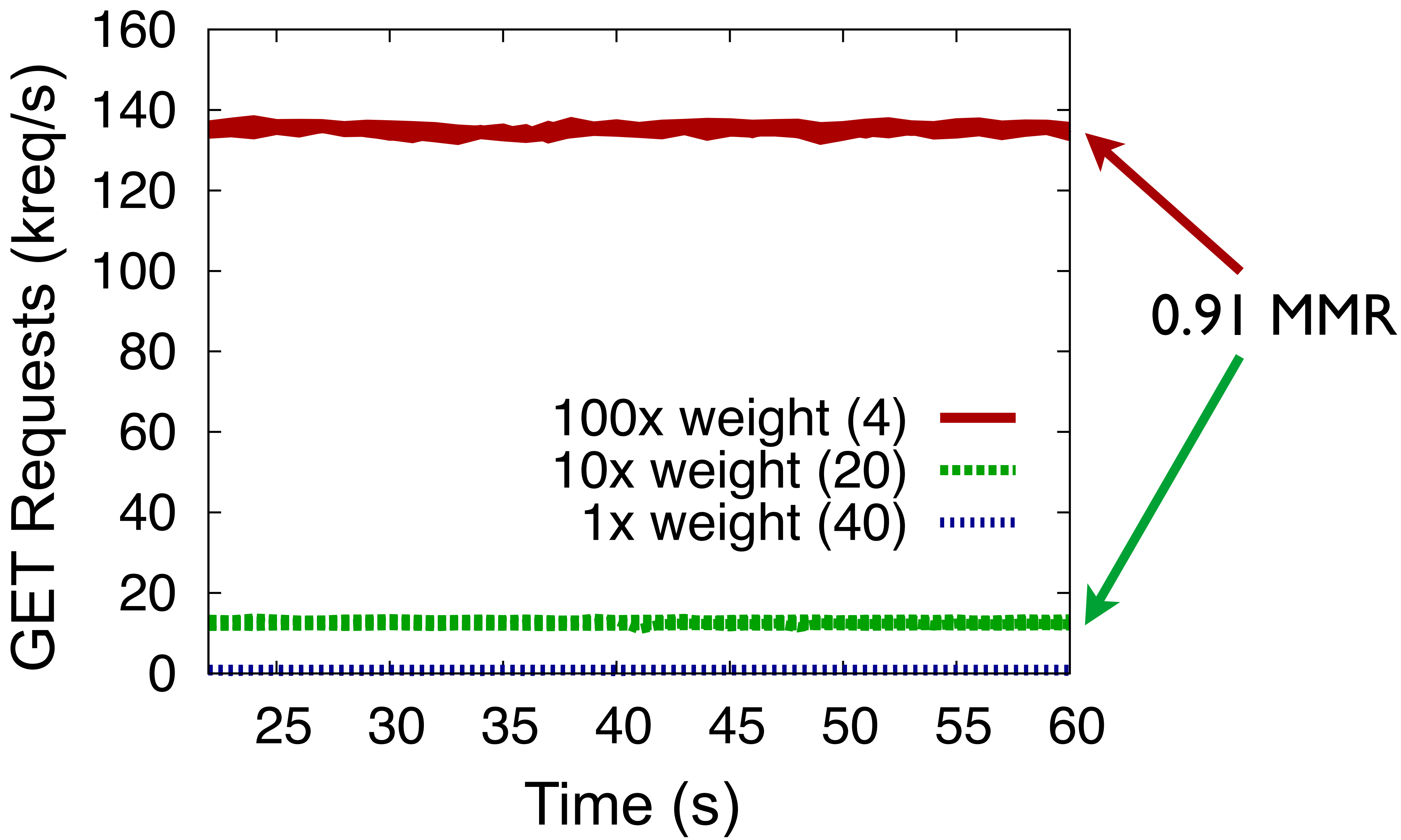


Pisces Achieves System-wide Weighted Fairness

0.98 MMR
4 heavy hitters

0.89 MMR
20 moderate demand

0.91 MMR
40 low demand

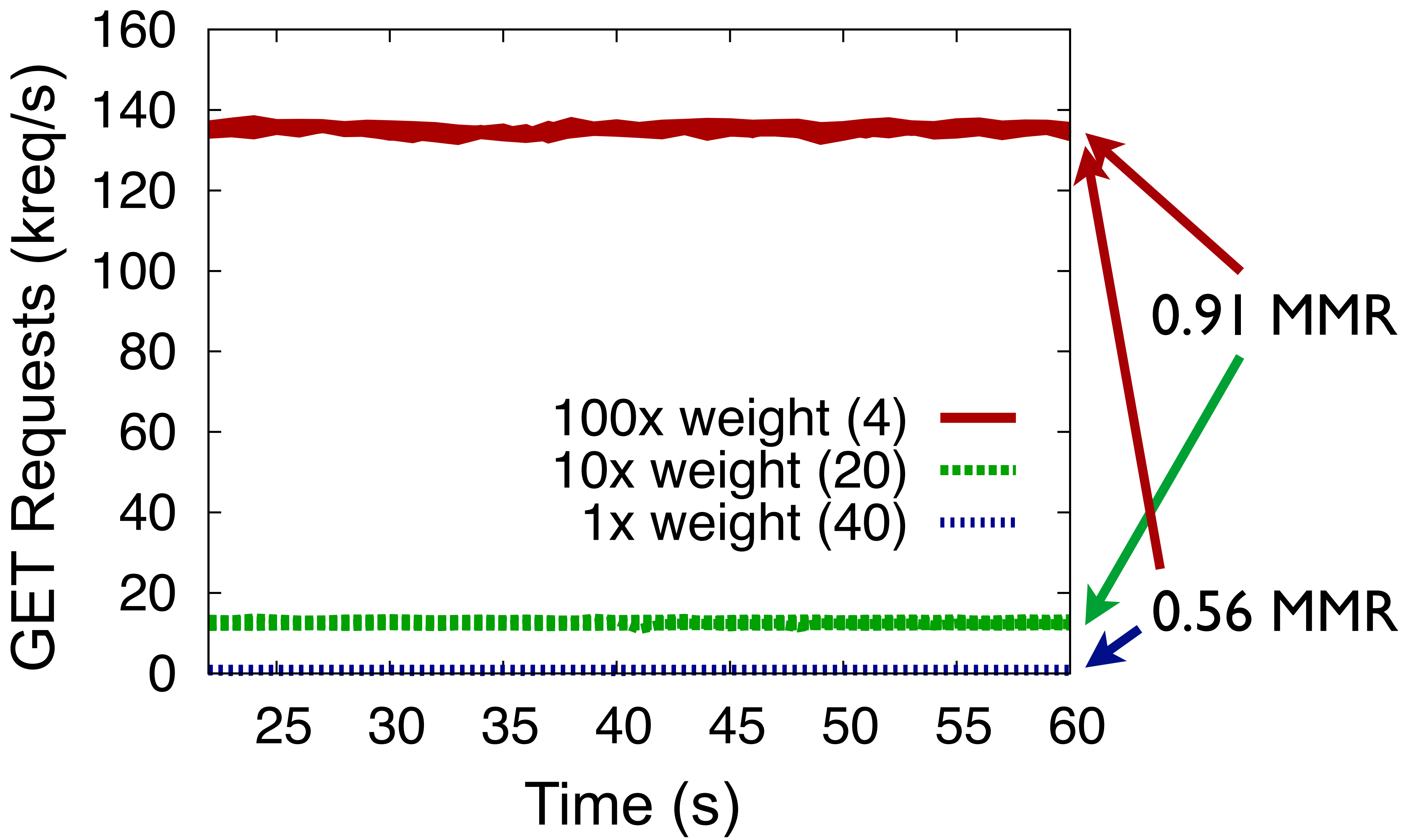


Pisces Achieves System-wide Weighted Fairness

0.98 MMR
4 heavy hitters

0.89 MMR
20 moderate demand

0.91 MMR
40 low demand



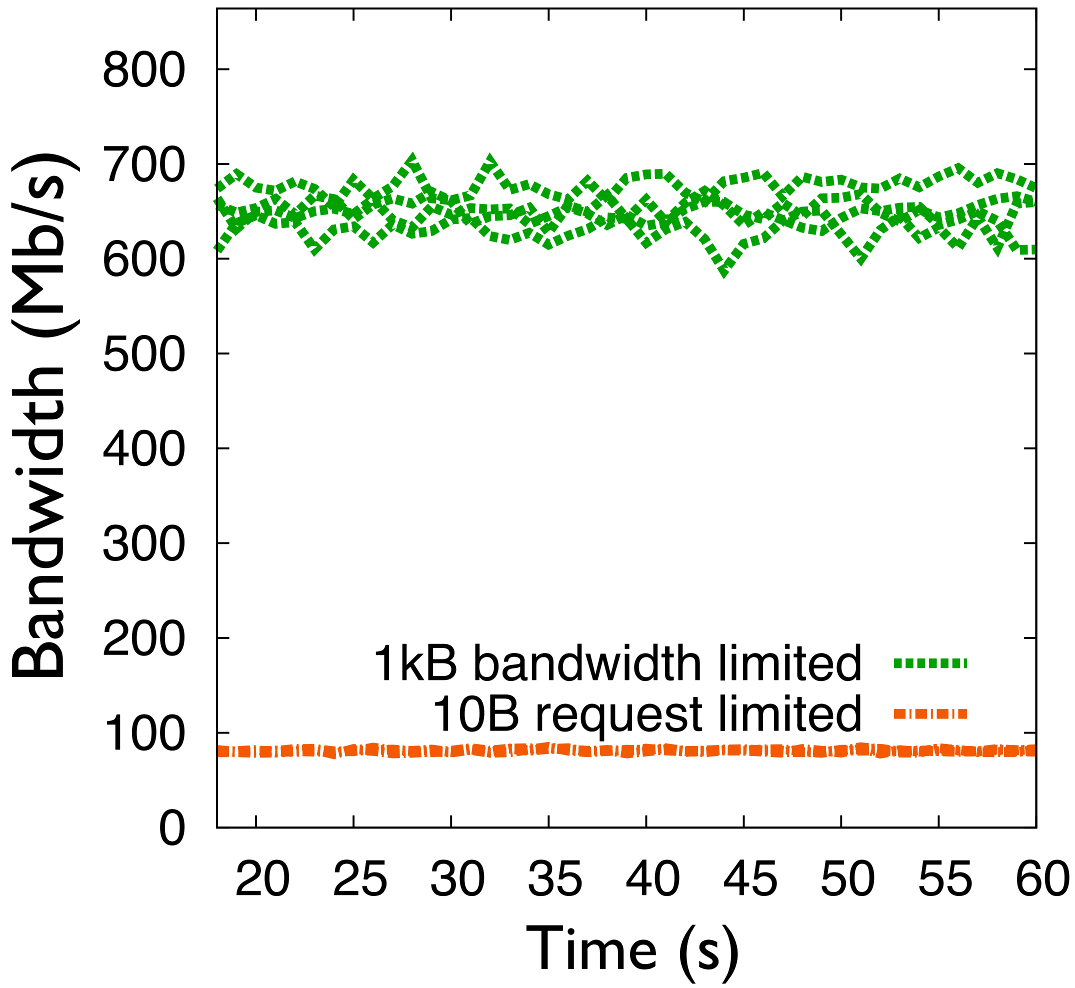
Pisces Achieves Dominant Resource Fairness

1kB workload
bandwidth limited

10B workload
request limited

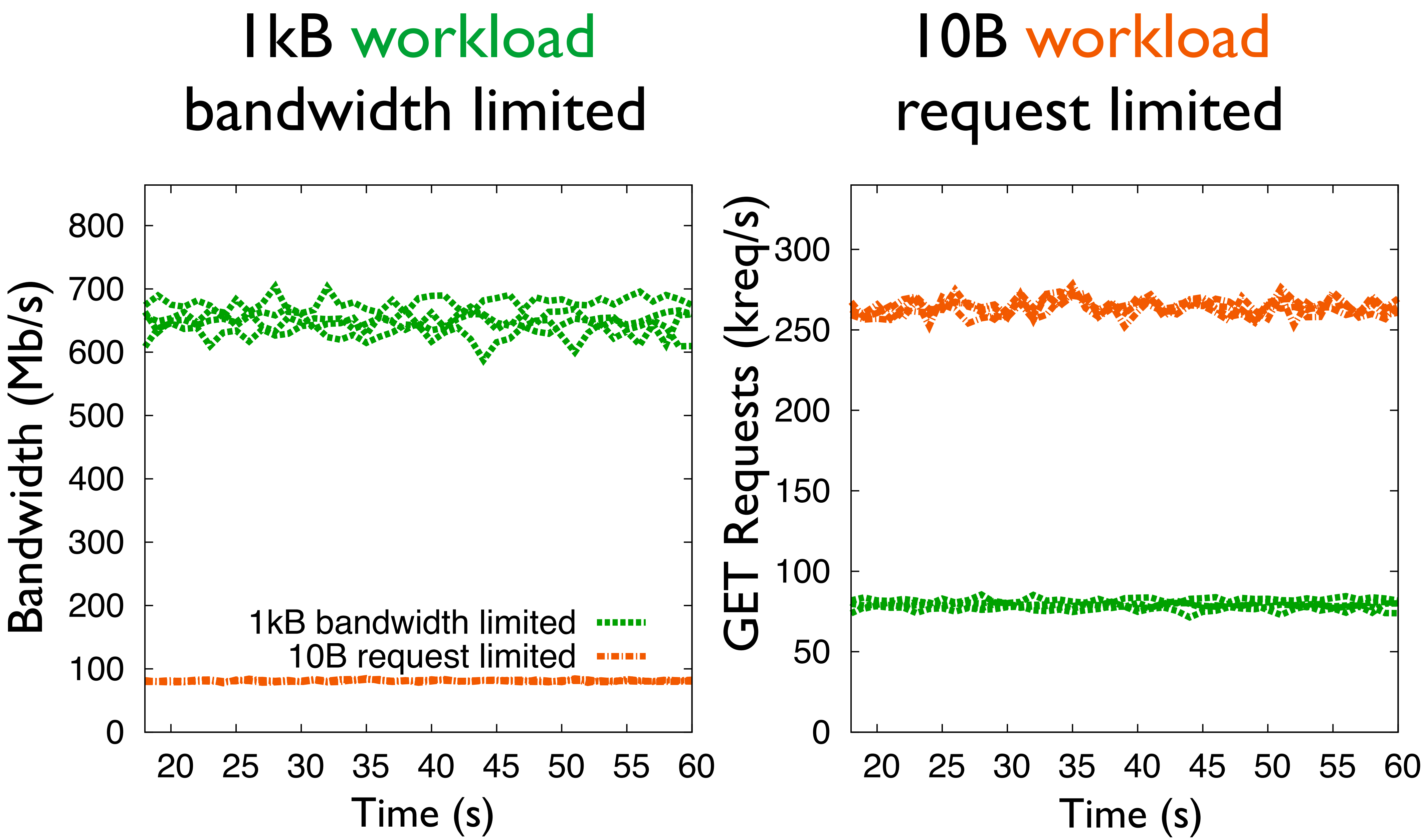
Pisces Achieves Dominant Resource Fairness

1kB workload
bandwidth limited

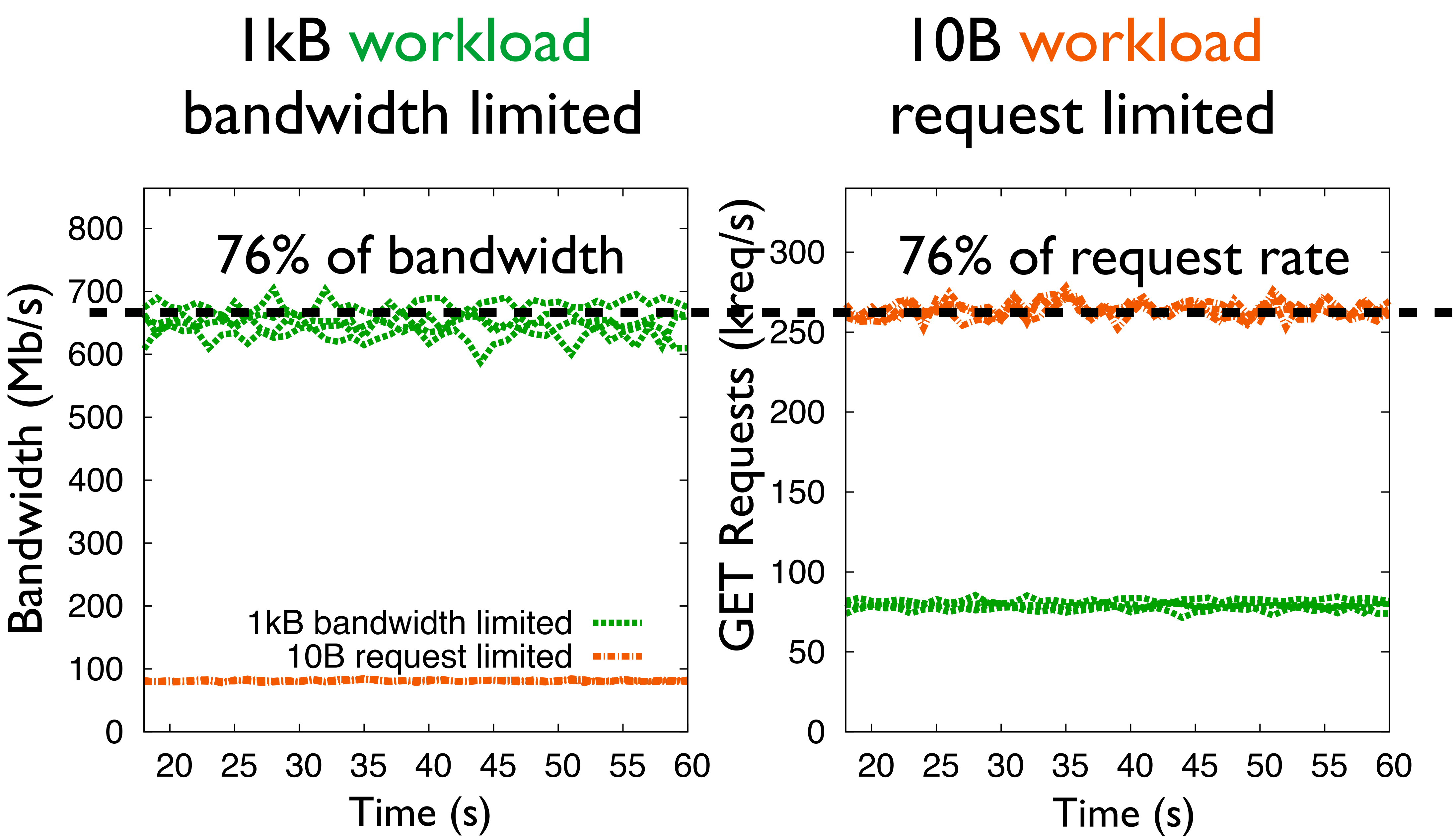


10B workload
request limited

Pisces Achieves Dominant Resource Fairness



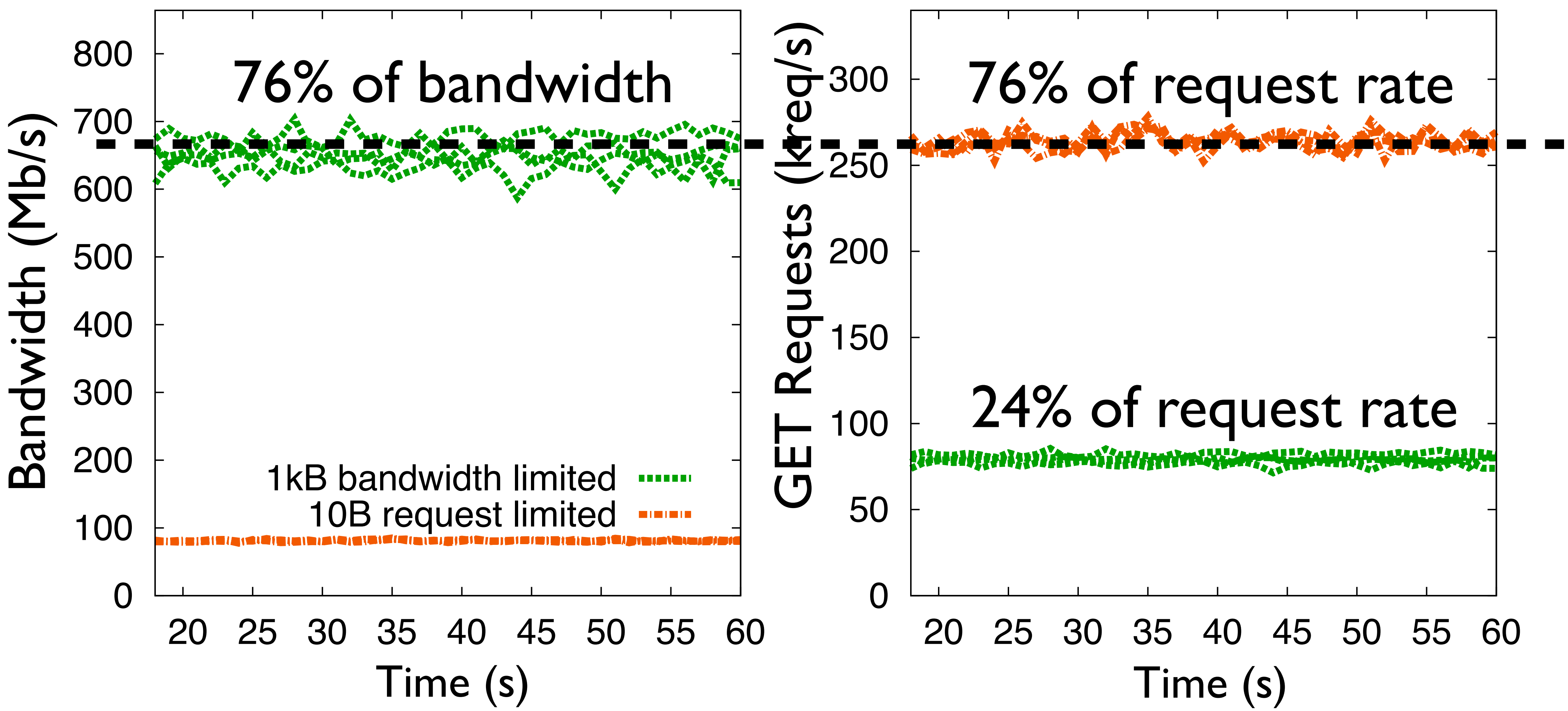
Pisces Achieves Dominant Resource Fairness



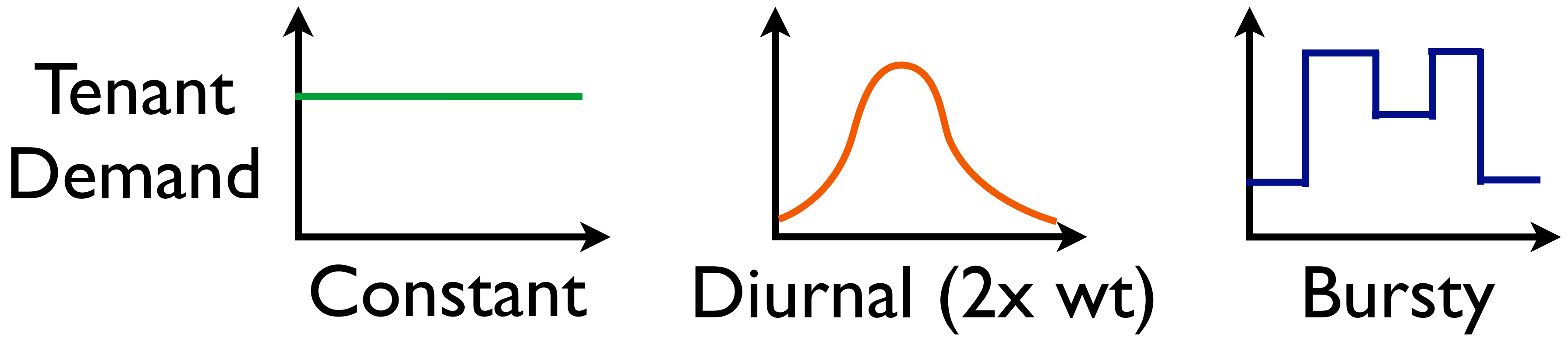
Pisces Achieves Dominant Resource Fairness

1kB workload
bandwidth limited

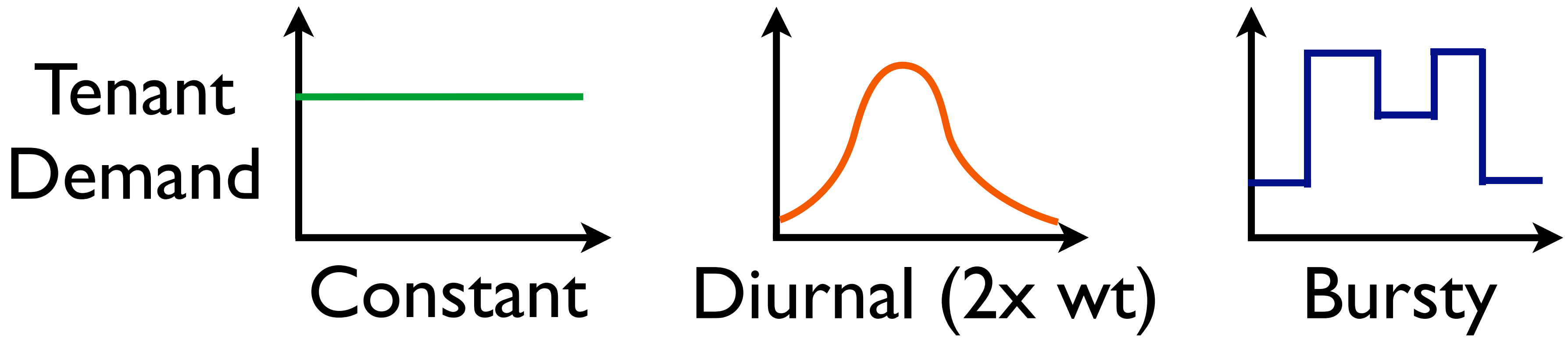
10B workload
request limited

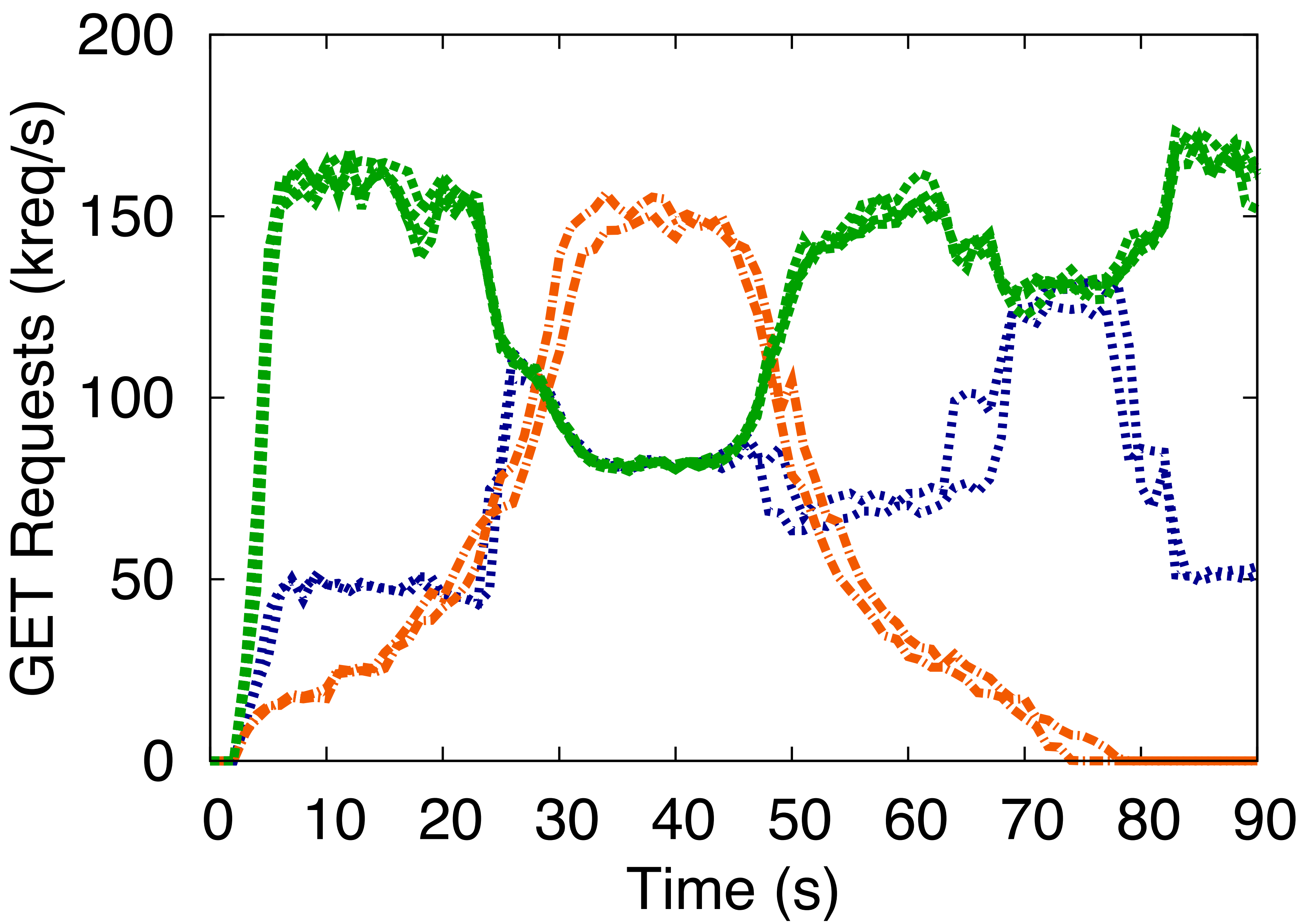


Pisces Adapts to Dynamic Demand

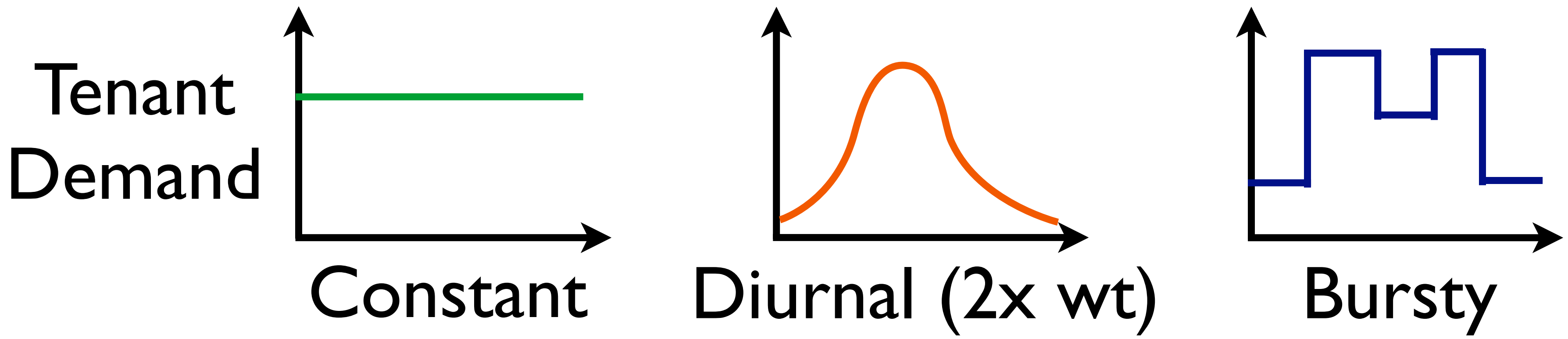


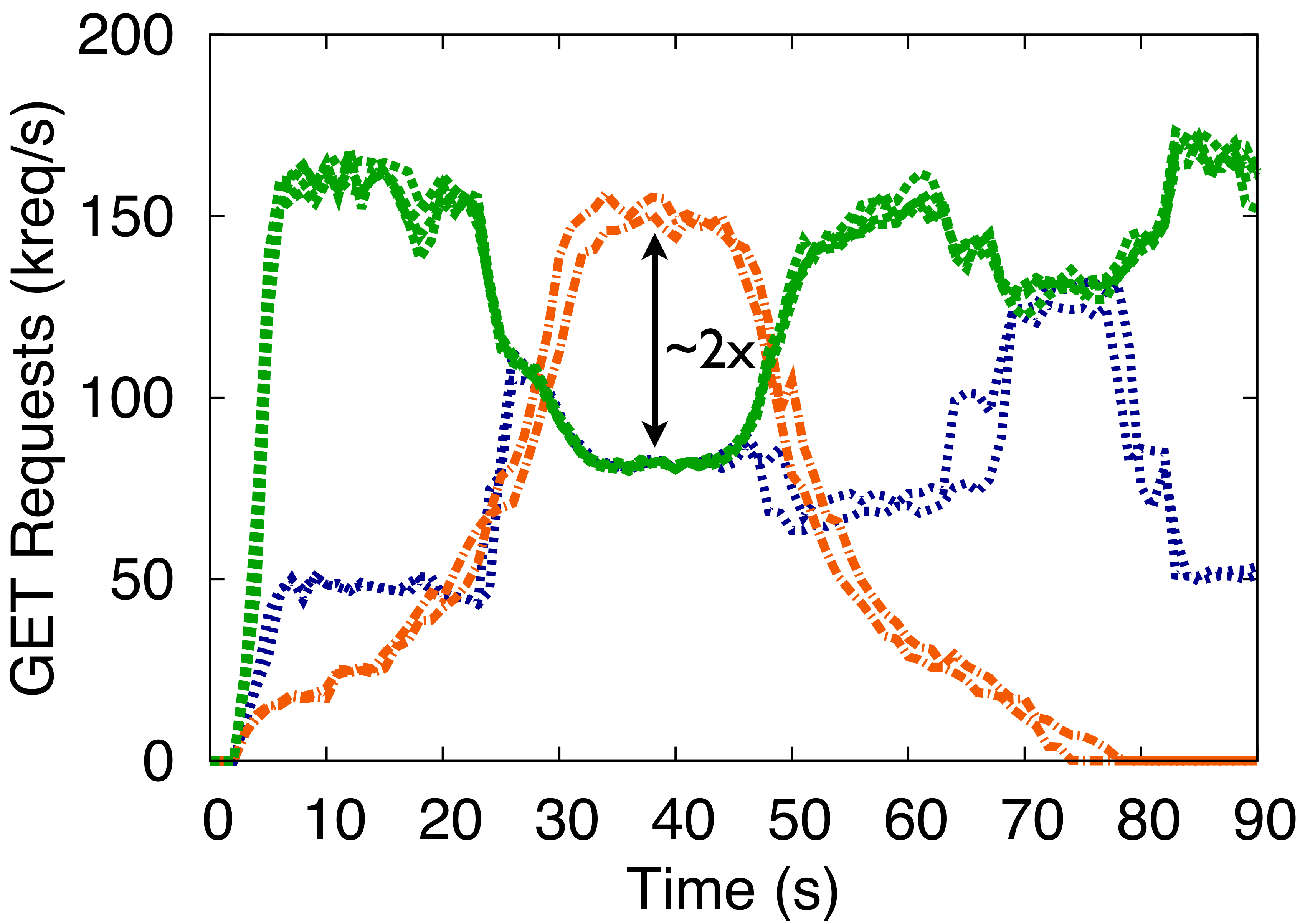
Pisces Adapts to Dynamic Demand



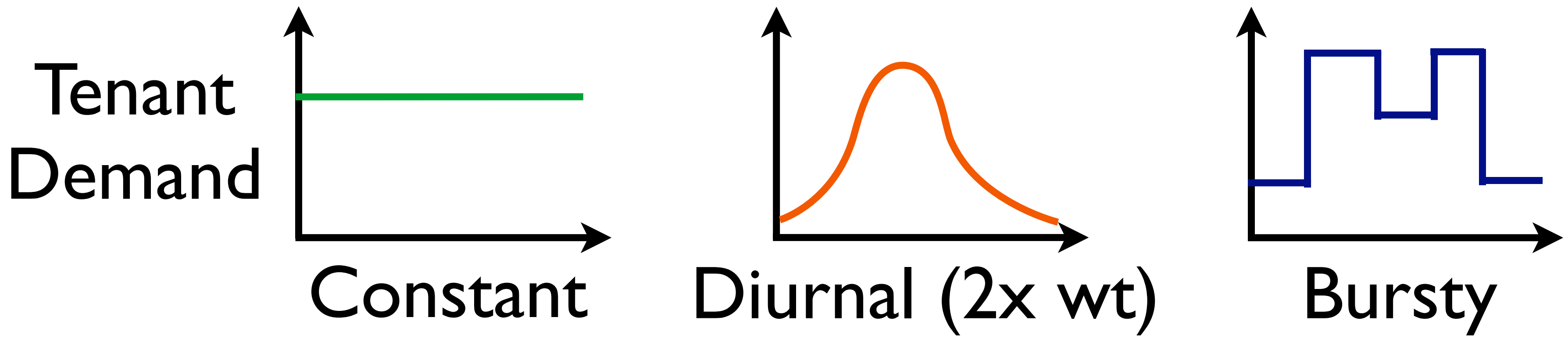


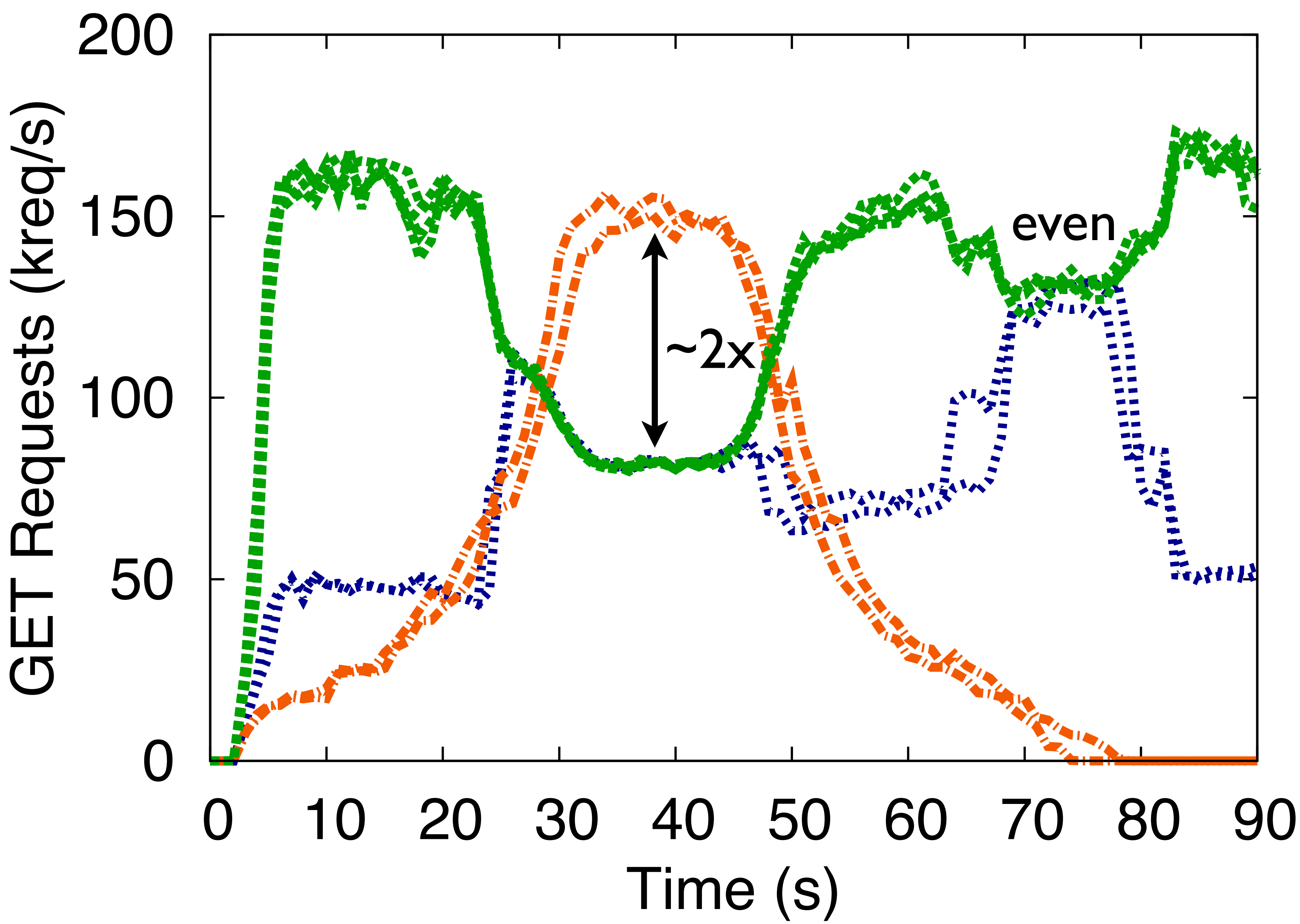
Pisces Adapts to Dynamic Demand





Pisces Adapts to Dynamic Demand





Conclusion

- Pisces Contributions
  - Per-tenant weighted max-min fair shares of system-wide resources w/ high utilization
  - Arbitrary object distributions
  - Different resource bottlenecks
  - Novel decomposition into 4 complementary mechanisms

PP Partition Placement | WA Weight Allocation | RS Replica Selection | FQ Fair Queuing