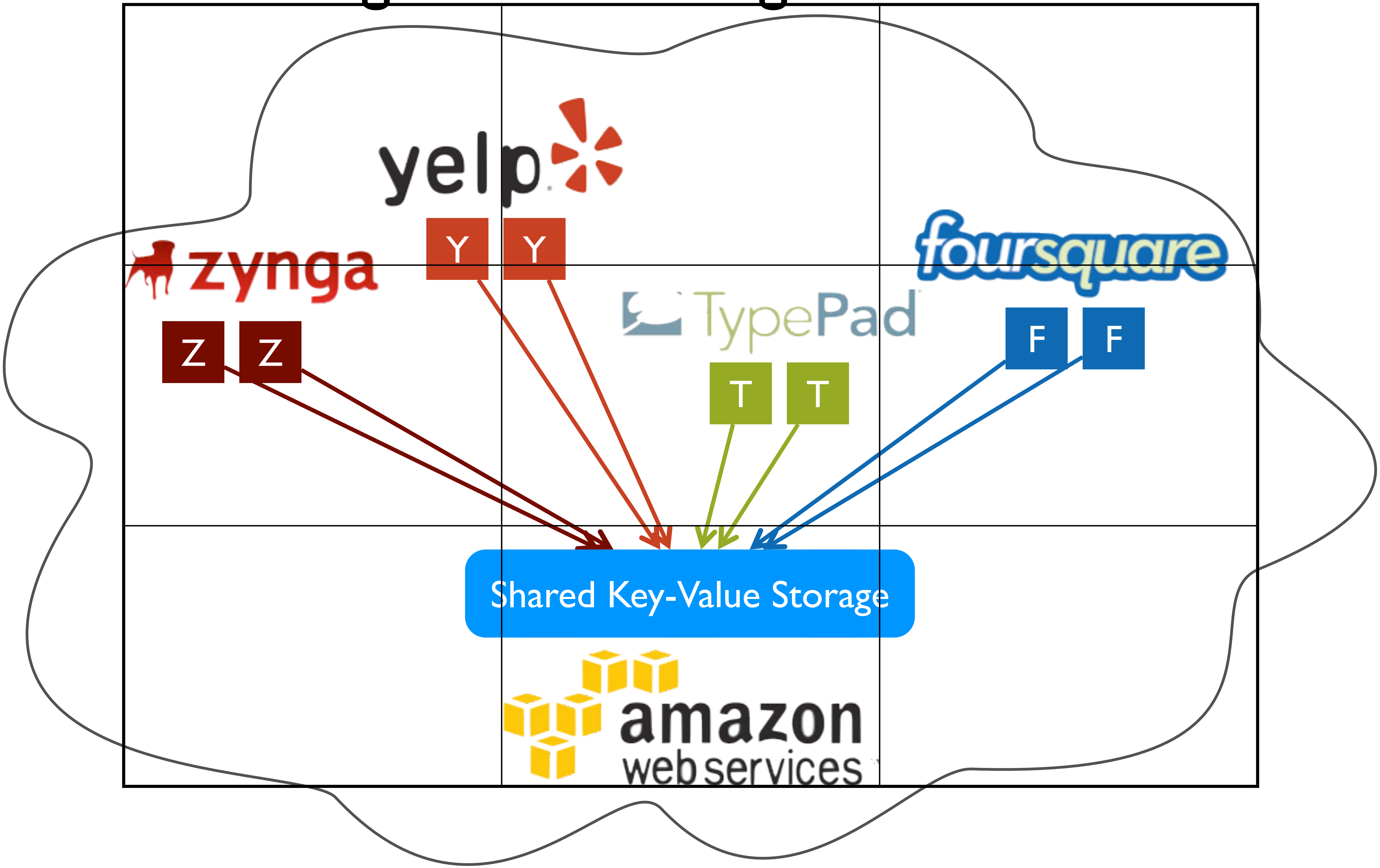
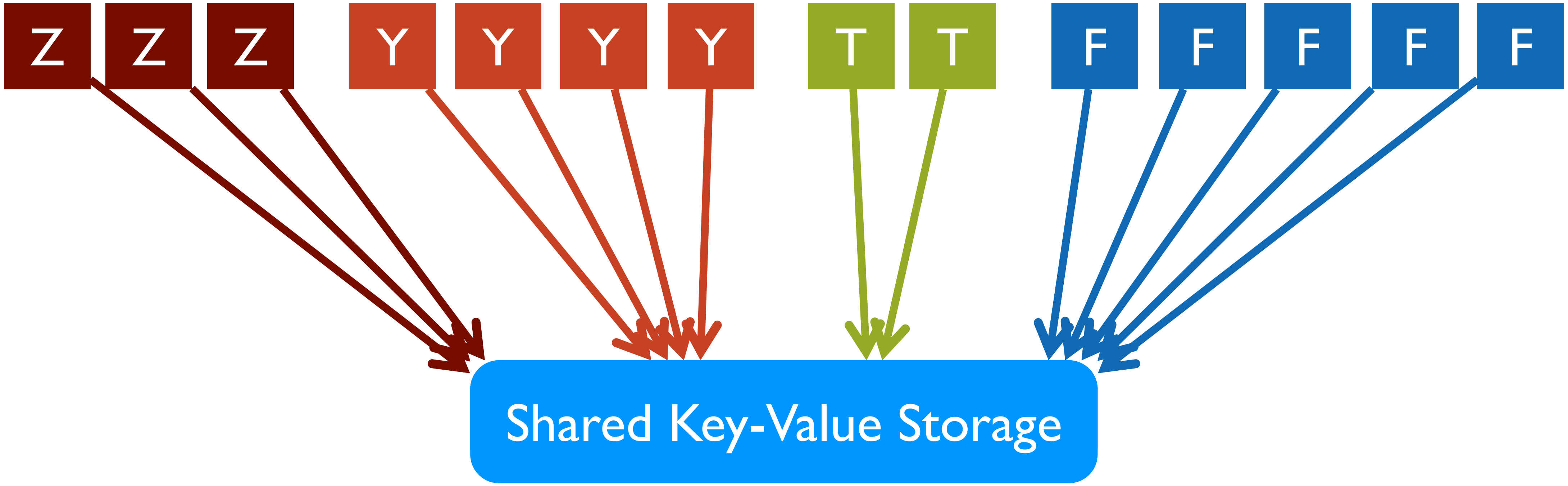
Performance Isolation and Fairness for Multi-Tenant Cloud Storage

Zhang Zhizhong   Oct 17, 2012

Setting: Shared Storage in the Cloud

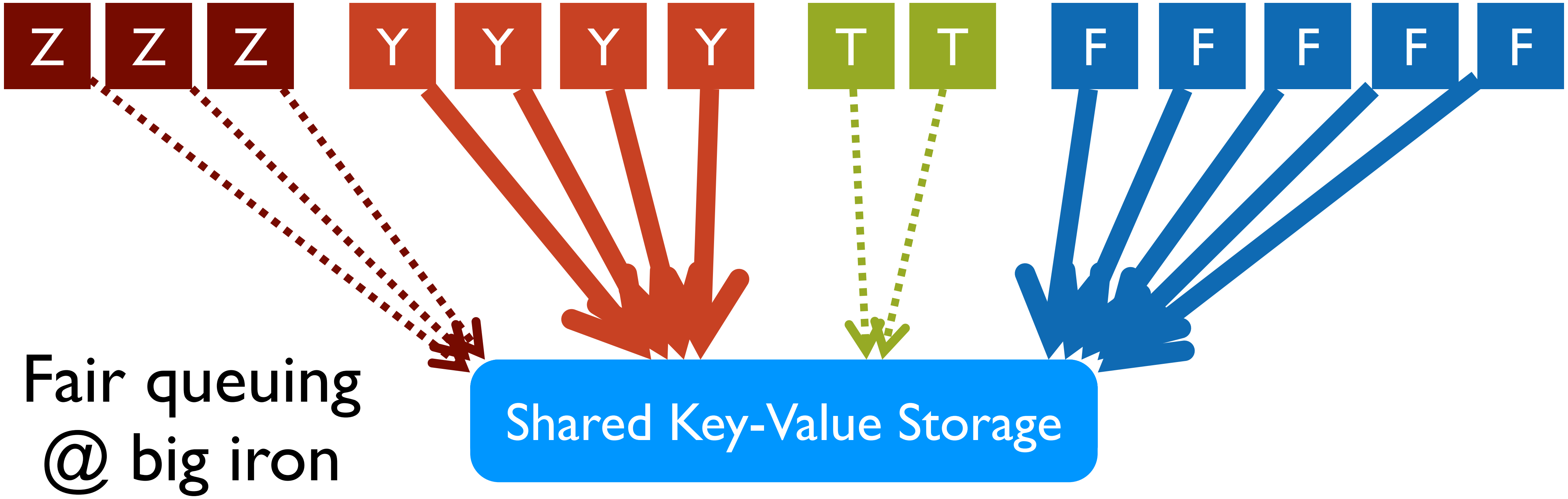


Predictable Performance is Hard



Multiple co-located tenants ⇒ resource contention

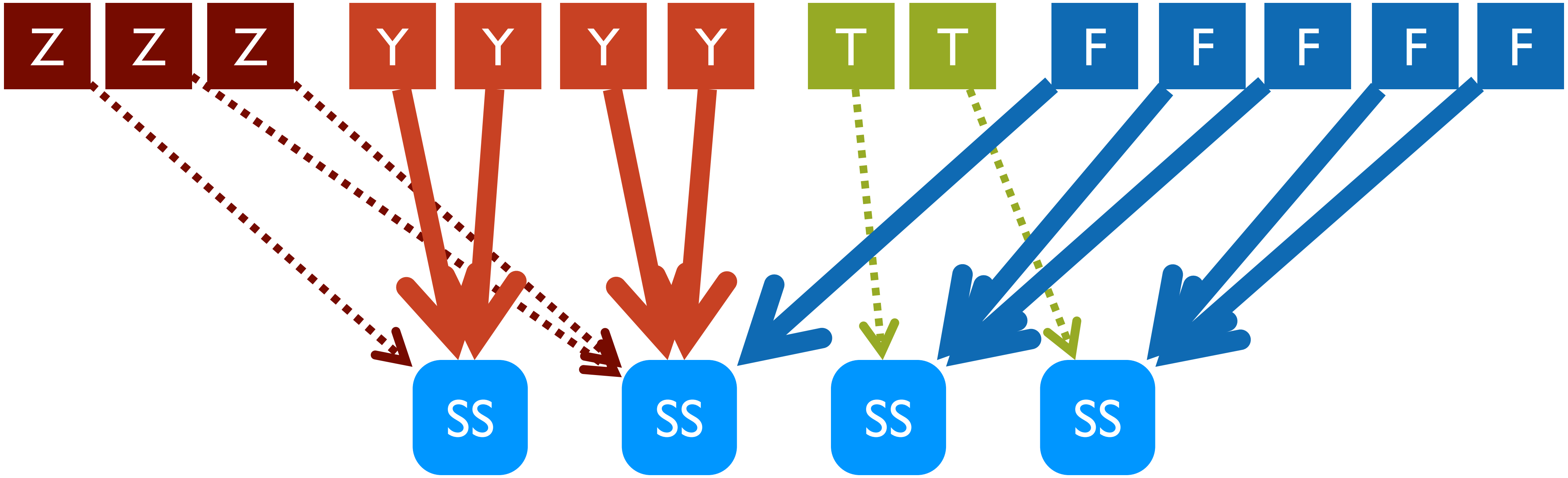
Predictable Performance is Hard



Fair queuing
@ big iron

Multiple co-located tenants ⇒ resource contention

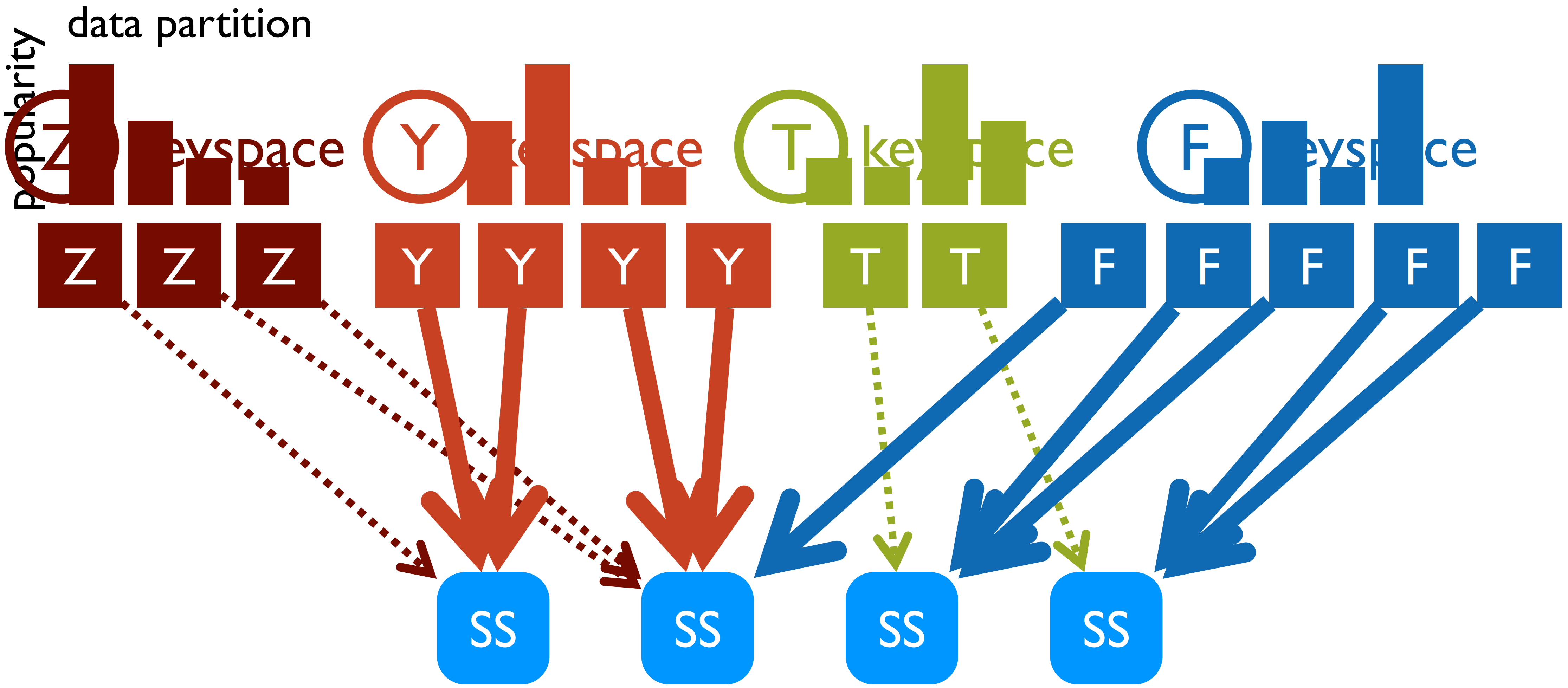
Predictable Performance is Hard



Multiple co-located tenants ⇒ resource contention

Distributed system ⇒ distributed resource allocation

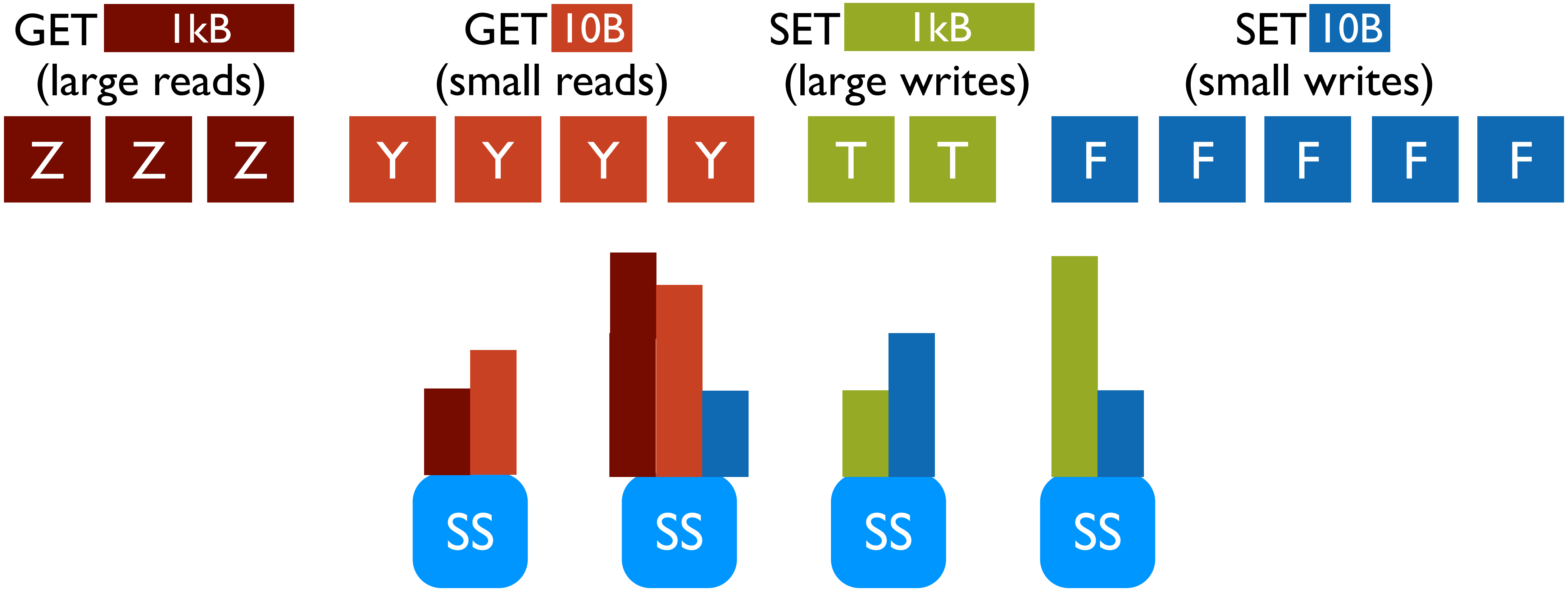
Predictable Performance is Hard



Multiple co-located tenants ⇒ resource contention

Distributed system ⇒ distributed resource allocation

Predictable Performance is Hard



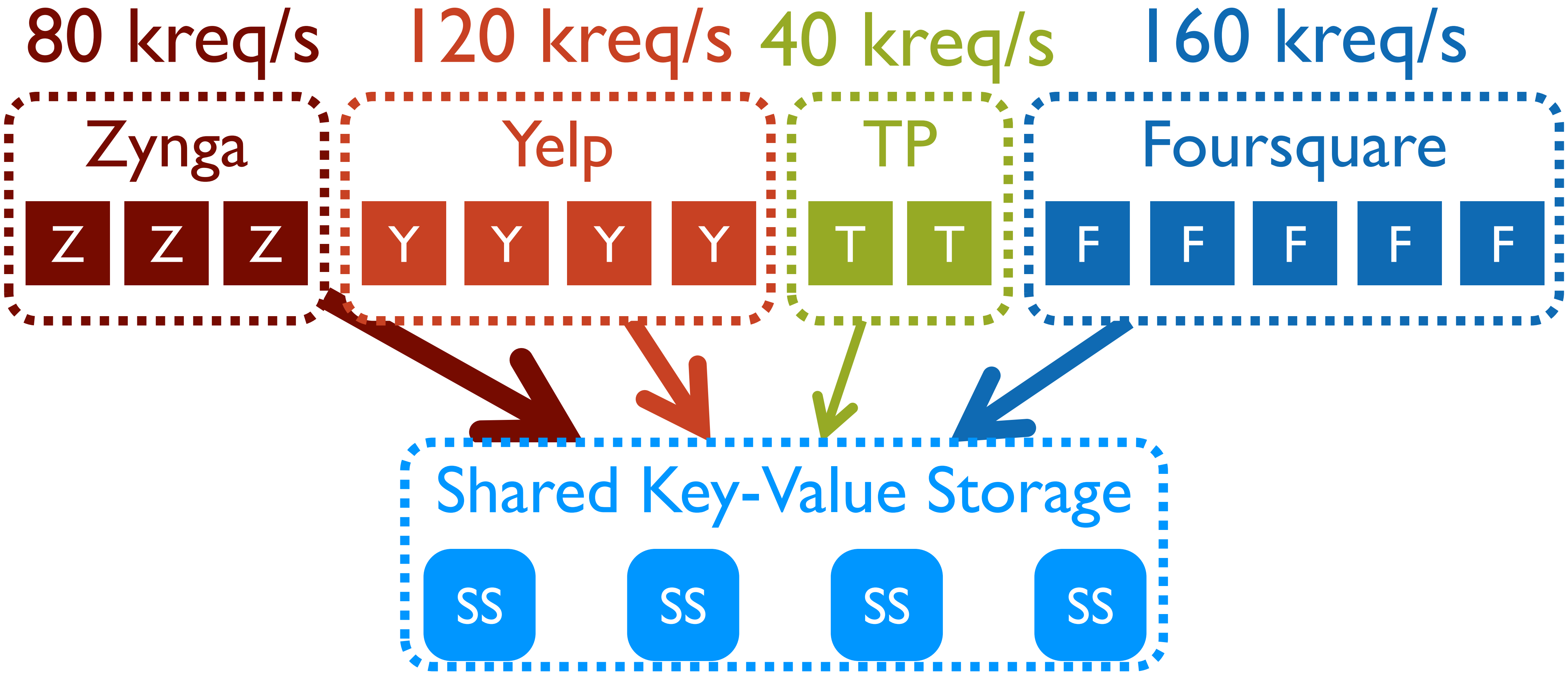
Multiple co-located tenants ⇒ resource contention

Distributed system ⇒ distributed resource allocation

Skewed object popularity ⇒ variable per-node demand

Disparate workloads ⇒ different bottleneck resources

Tenants Want System-wide Resource Guarantees



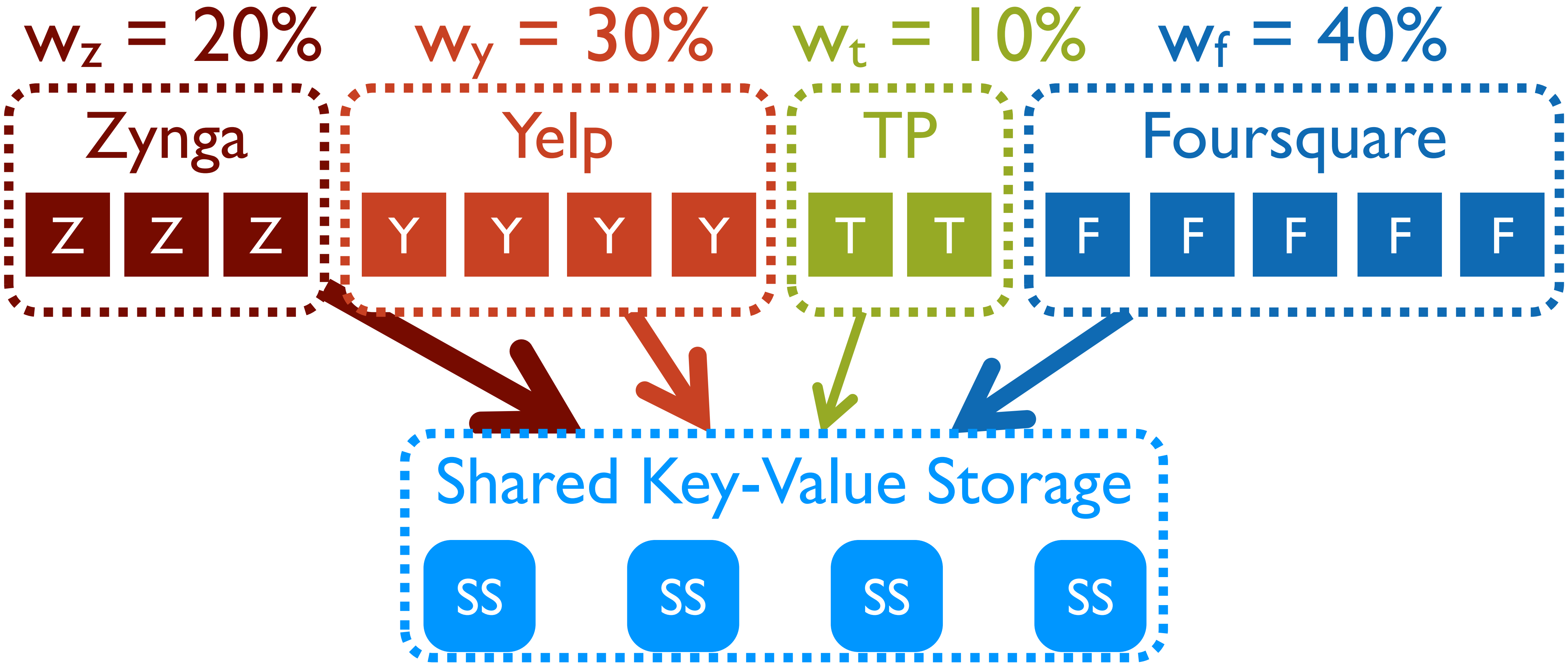
Multiple co-located tenants ⇒ resource contention

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

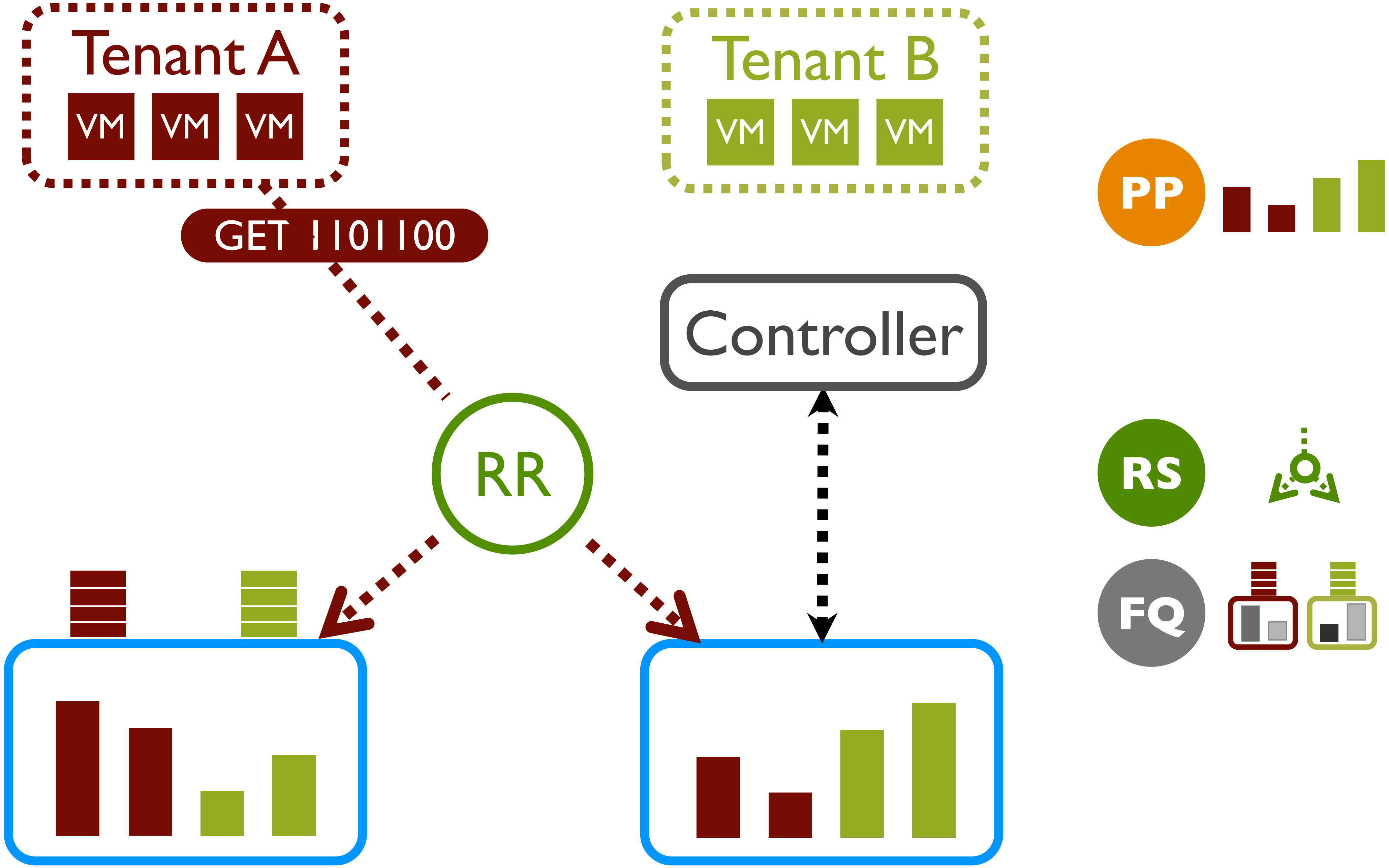
Skewed object popularity ⇒ variable per-node demand

Disparate workloads ⇒ different bottleneck resources

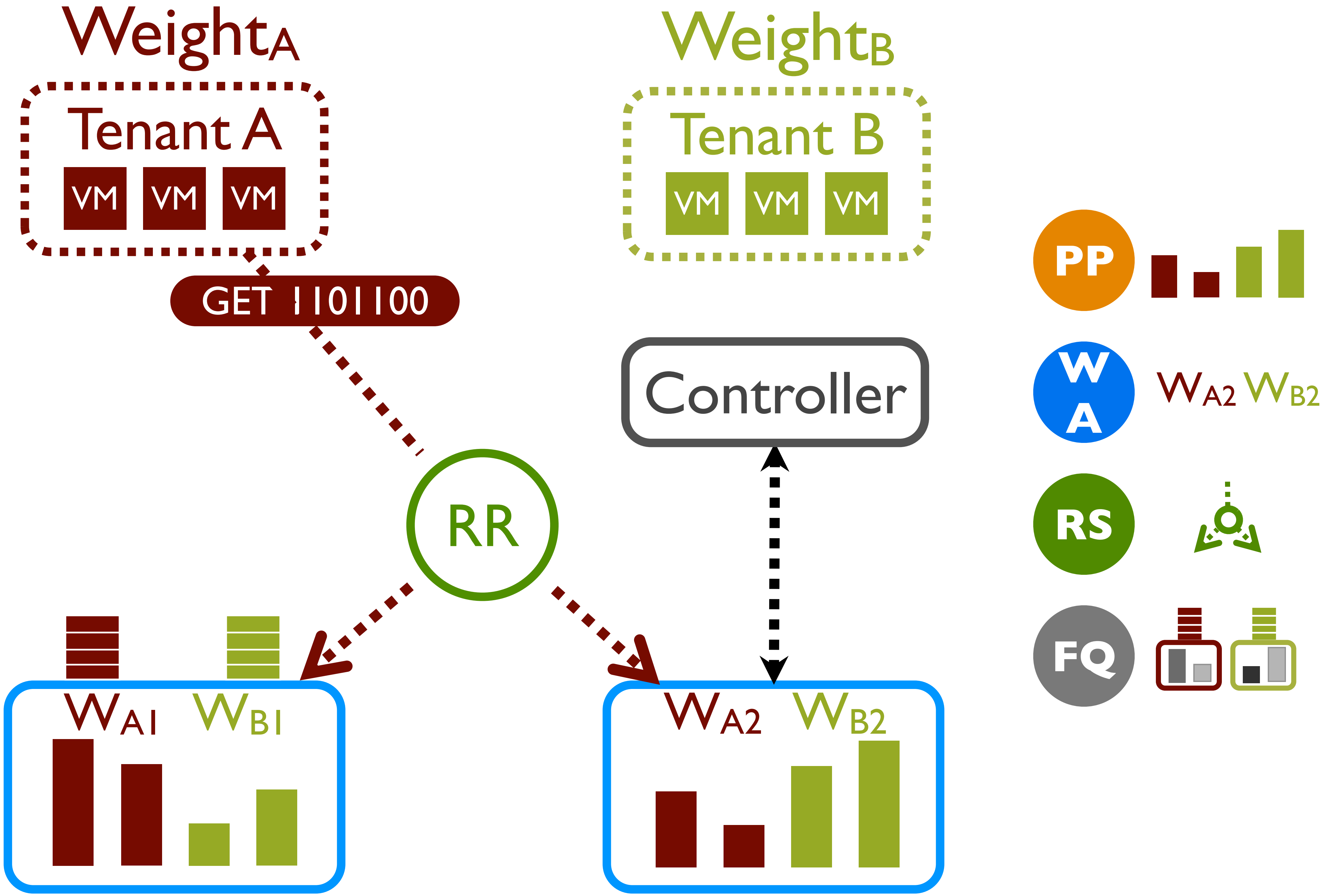
Pisces: Predictable Shared Cloud Storage

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

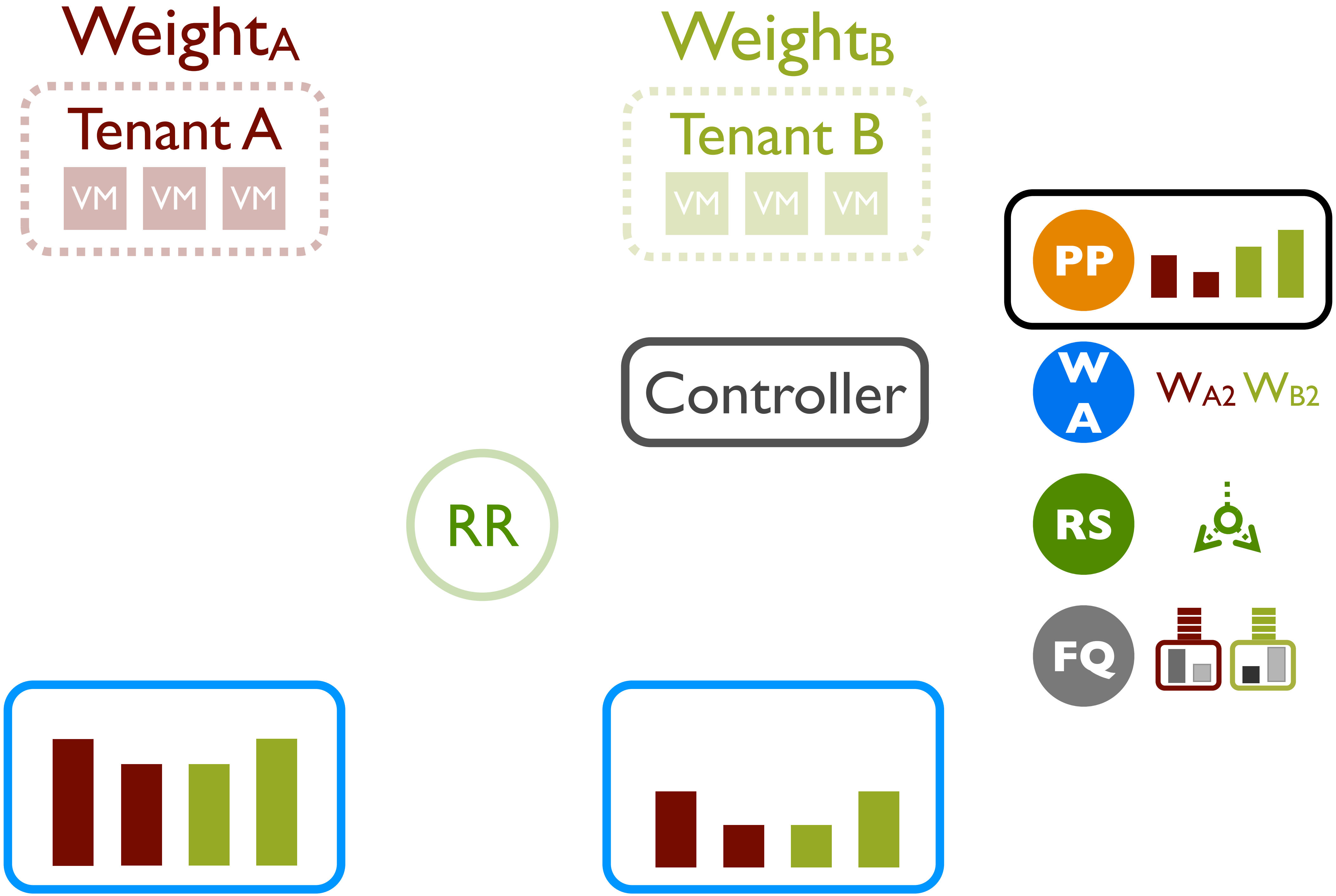
Predictable Multi-Tenant Key-Value Storage



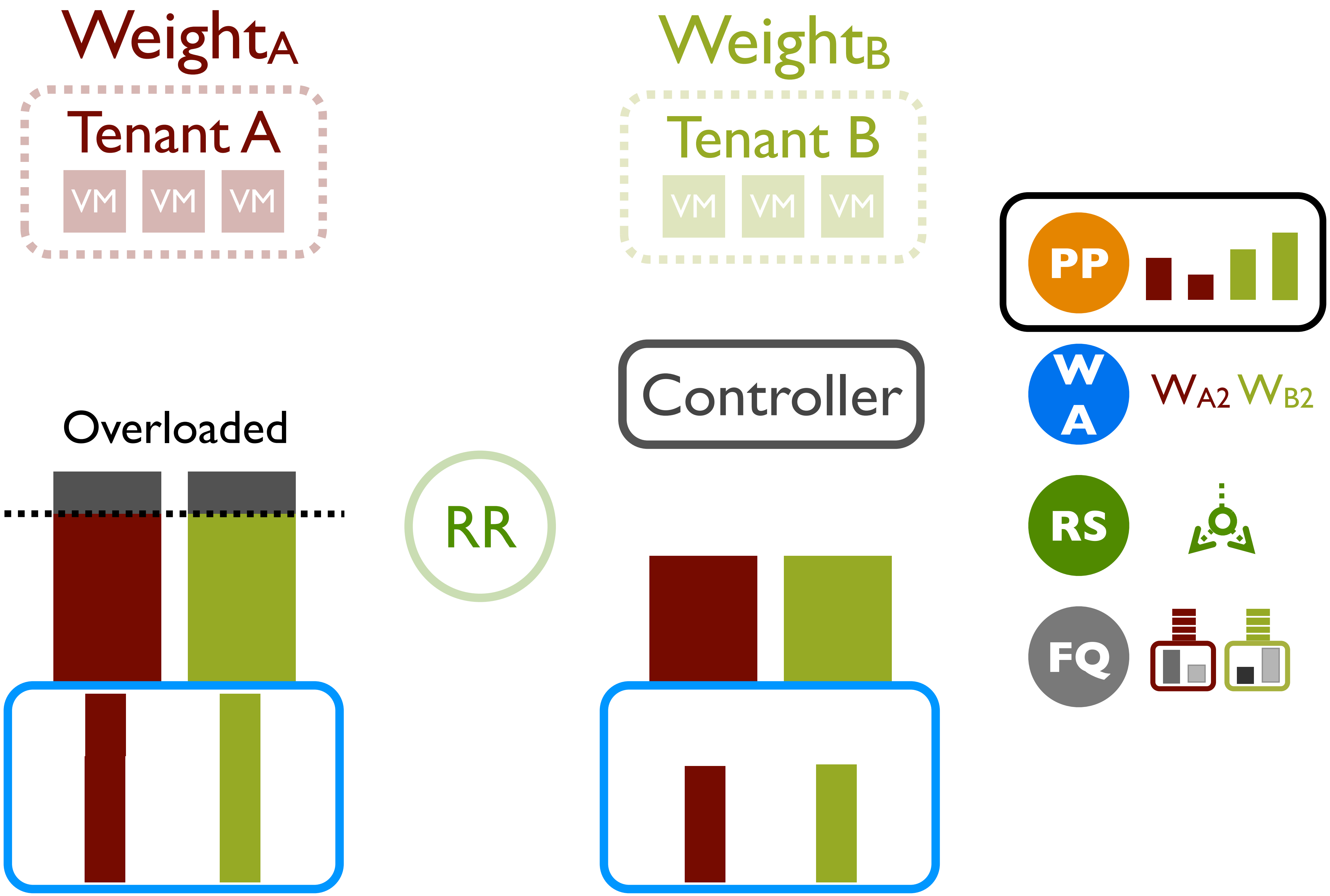
Predictable Multi-Tenant Key-Value Storage



Strawman: Place Partitions Randomly



Strawman: Place Partitions Randomly



Weight_A

Tenant A

VM VM VM

Weight_B

Tenant B

VM VM VM

Controller

Overloaded

RR

PP

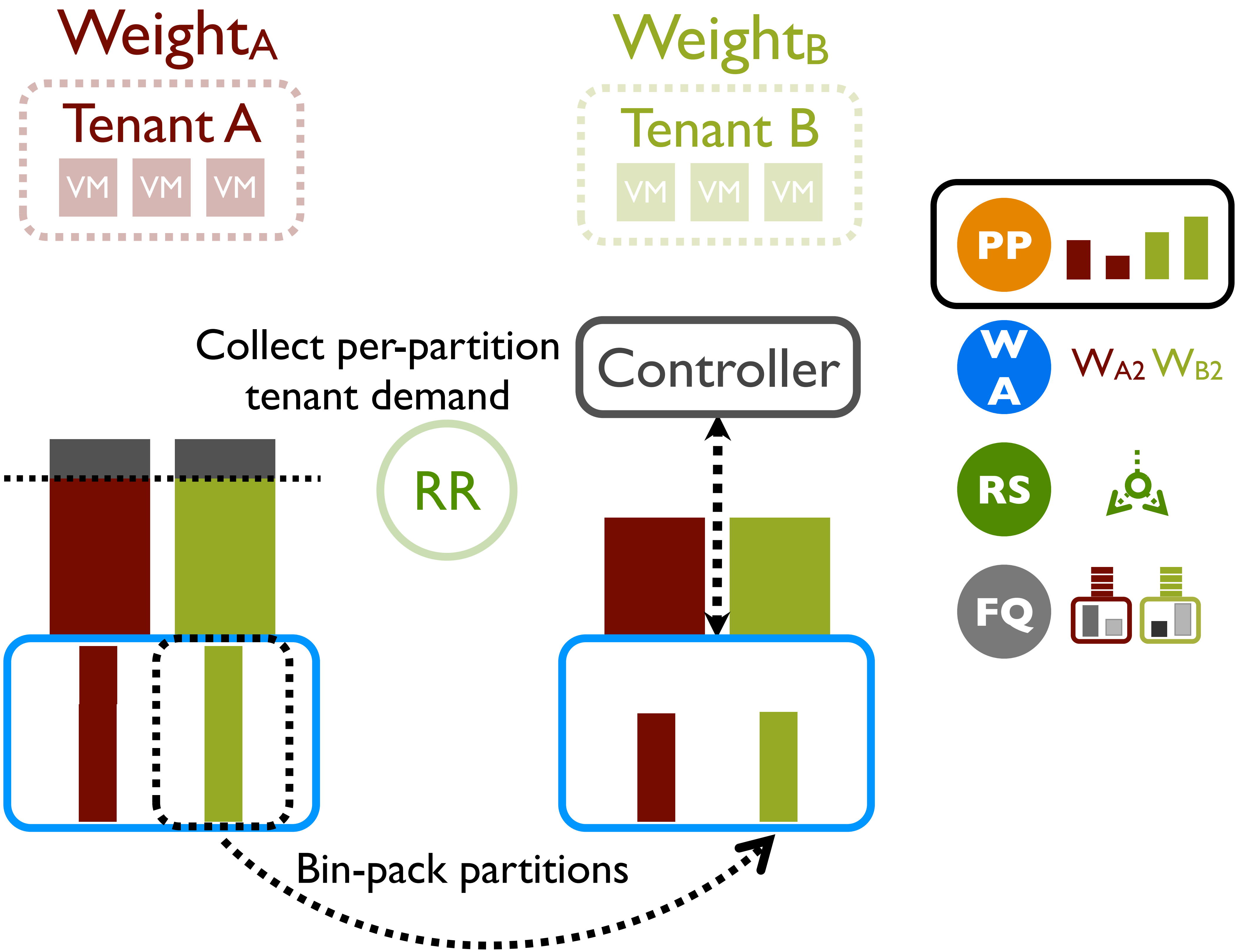
WA

W_{A2} W_{B2}

RS

FQ

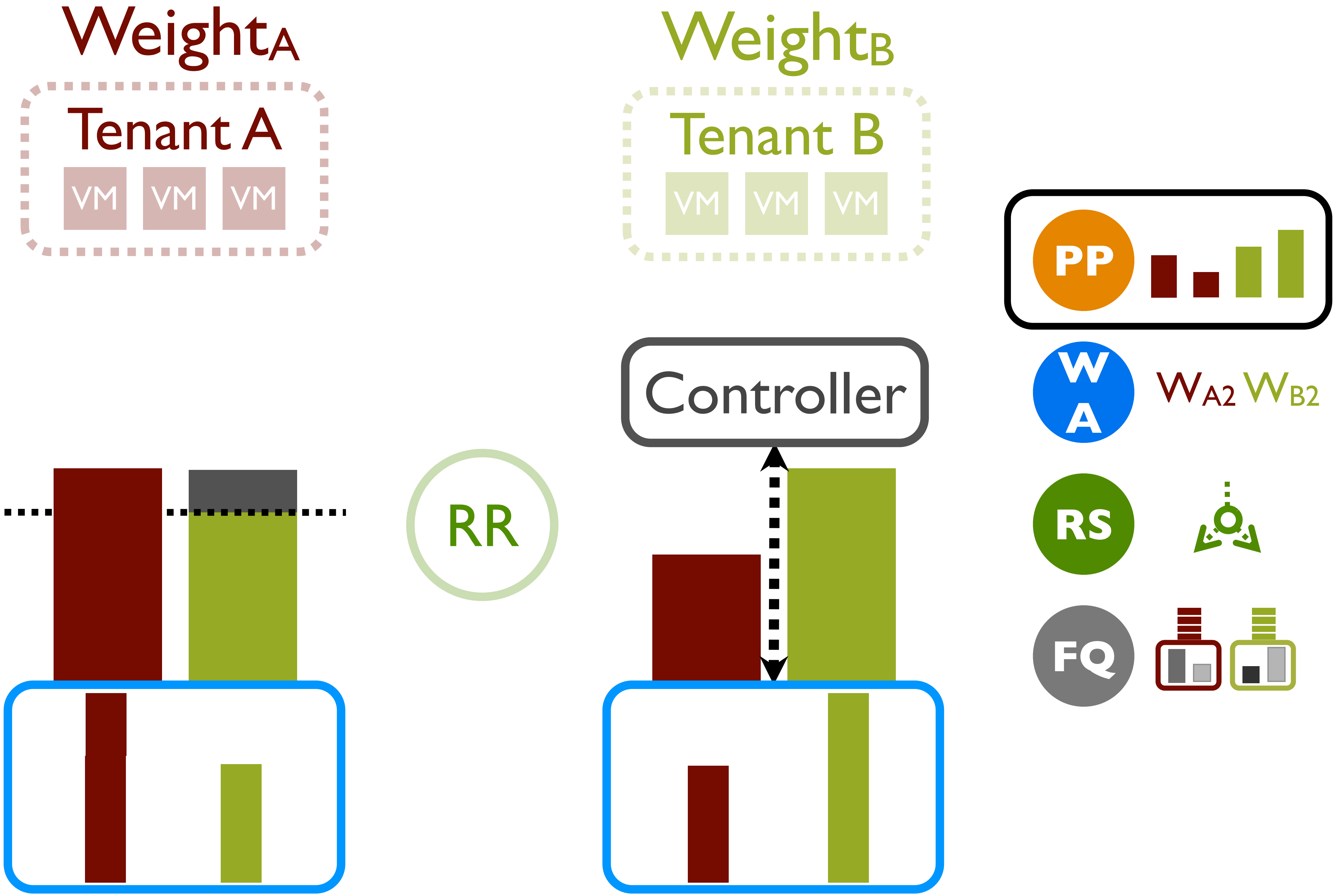
Pisces: Place Partitions By Fairness Constraints



Collect per-partition tenant demand

Bin-pack partitions

Pisces: Place Partitions By Fairness Constraints



Weight$_A$

Tenant A

VM VM VM

Weight$_B$

Tenant B

VM VM VM

Controller

RR

PP

W A

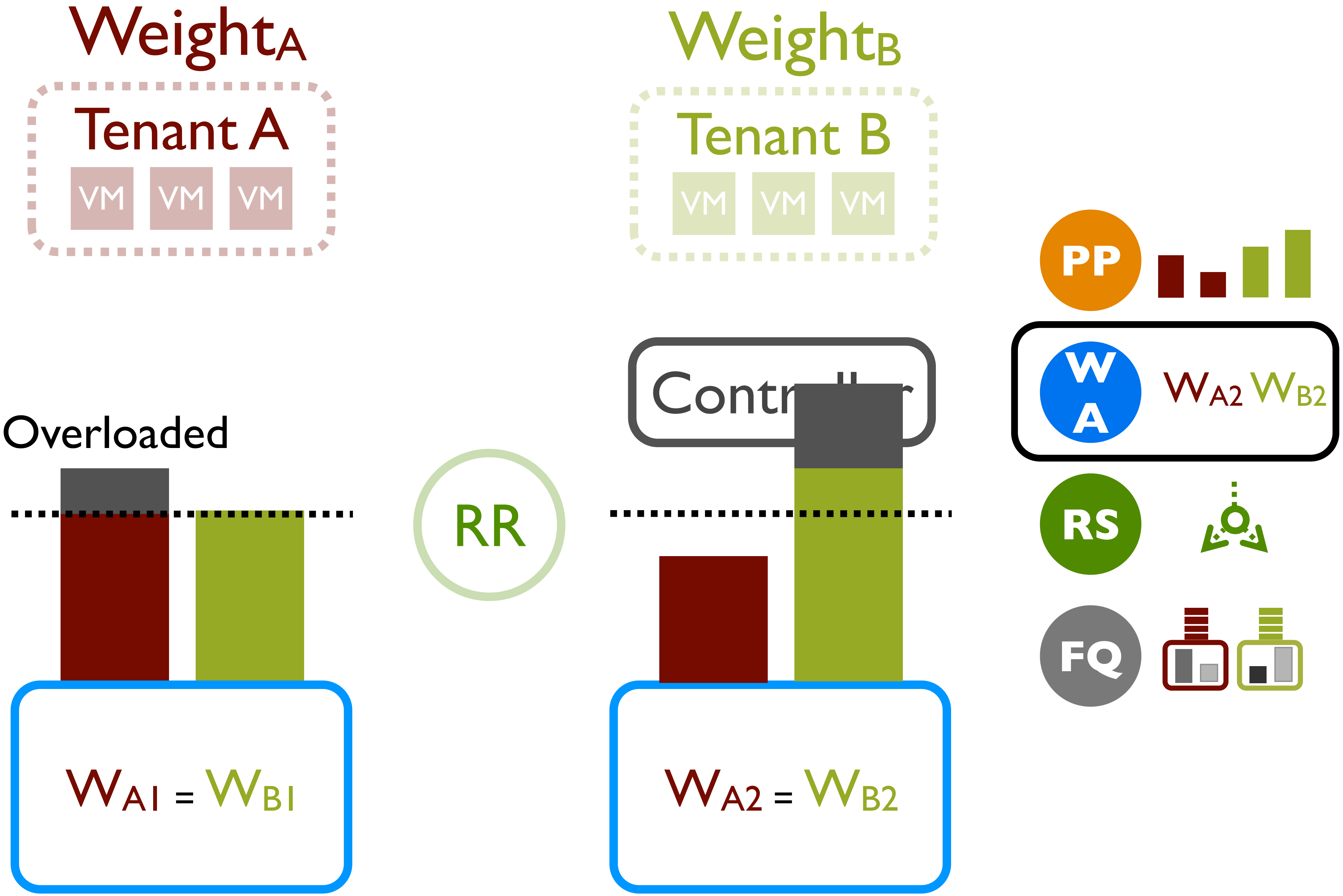
W$_{A2}$ W$_{B2}$

RS

FQ

Results in feasible partition placement

Strawman: Allocate Local Weights Evenly



$Weight_A$

Tenant A

VM VM VM

$Weight_B$

Tenant B

VM VM VM

Controller

Overloaded

RR

$W_{A1} = W_{B1}$

$W_{A2} = W_{B2}$

PP

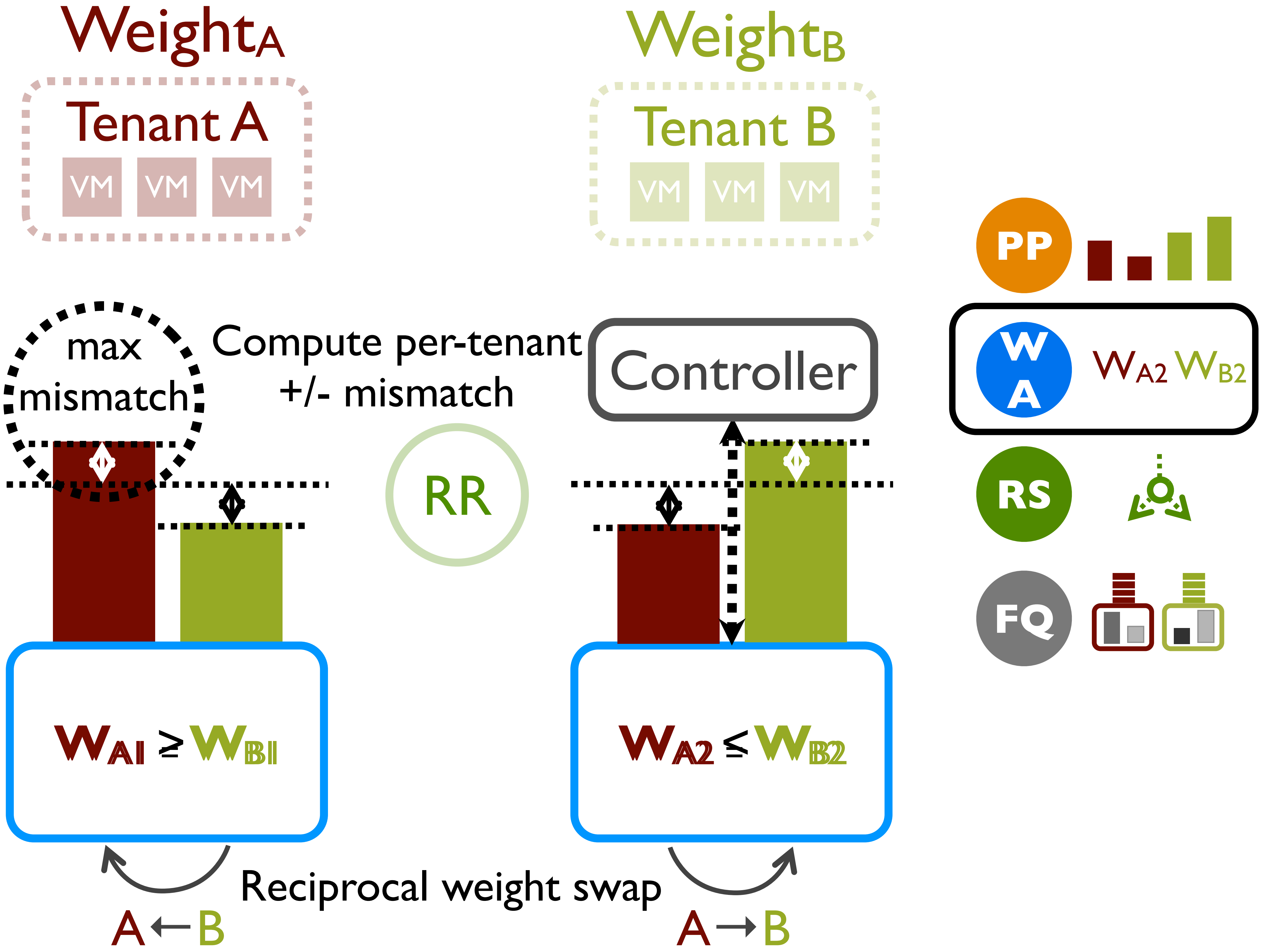
WA

W_{A2} W_{B2}

RS

FQ

Pisces: Allocate Local Weights By Tenant Demand



Weight_A

Tenant A

VM VM VM

Weight_B

Tenant B

VM VM VM

max mismatch

Compute per-tenant +/- mismatch

Controller

RR

$W_{A1} \geq W_{B1}$

$W_{A2} \leq W_{B2}$

Reciprocal weight swap

A ← B

A → B

PP

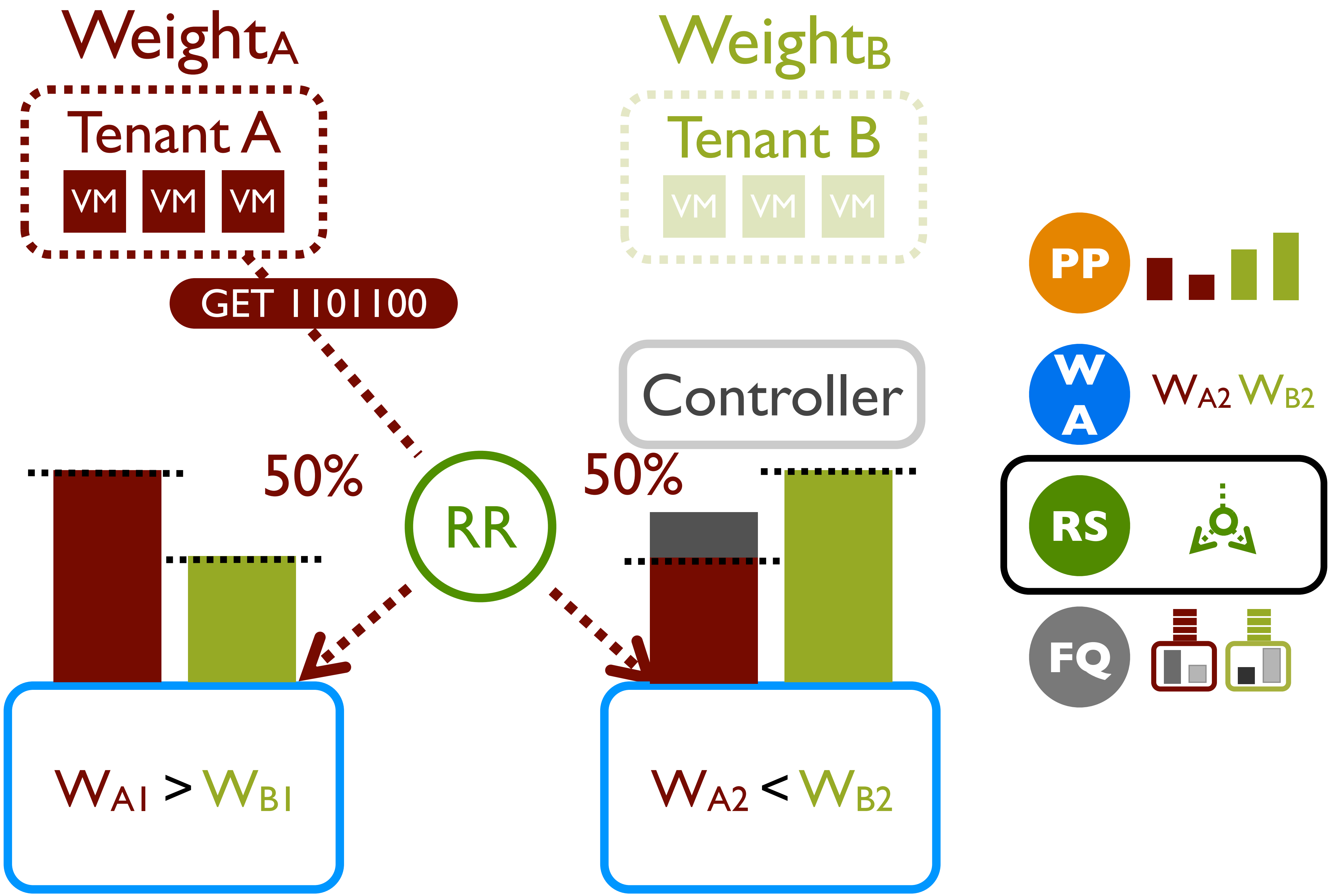
WA

W_{A2} W_{B2}

RS

FQ

Strawman: Select Replicas Evenly



Weight_A

Tenant A

VM VM VM

GET 1101100

Weight_B

Tenant B

VM VM VM

Controller

50%

RR

50%

$W_{A1} > W_{B1}$

$W_{A2} < W_{B2}$

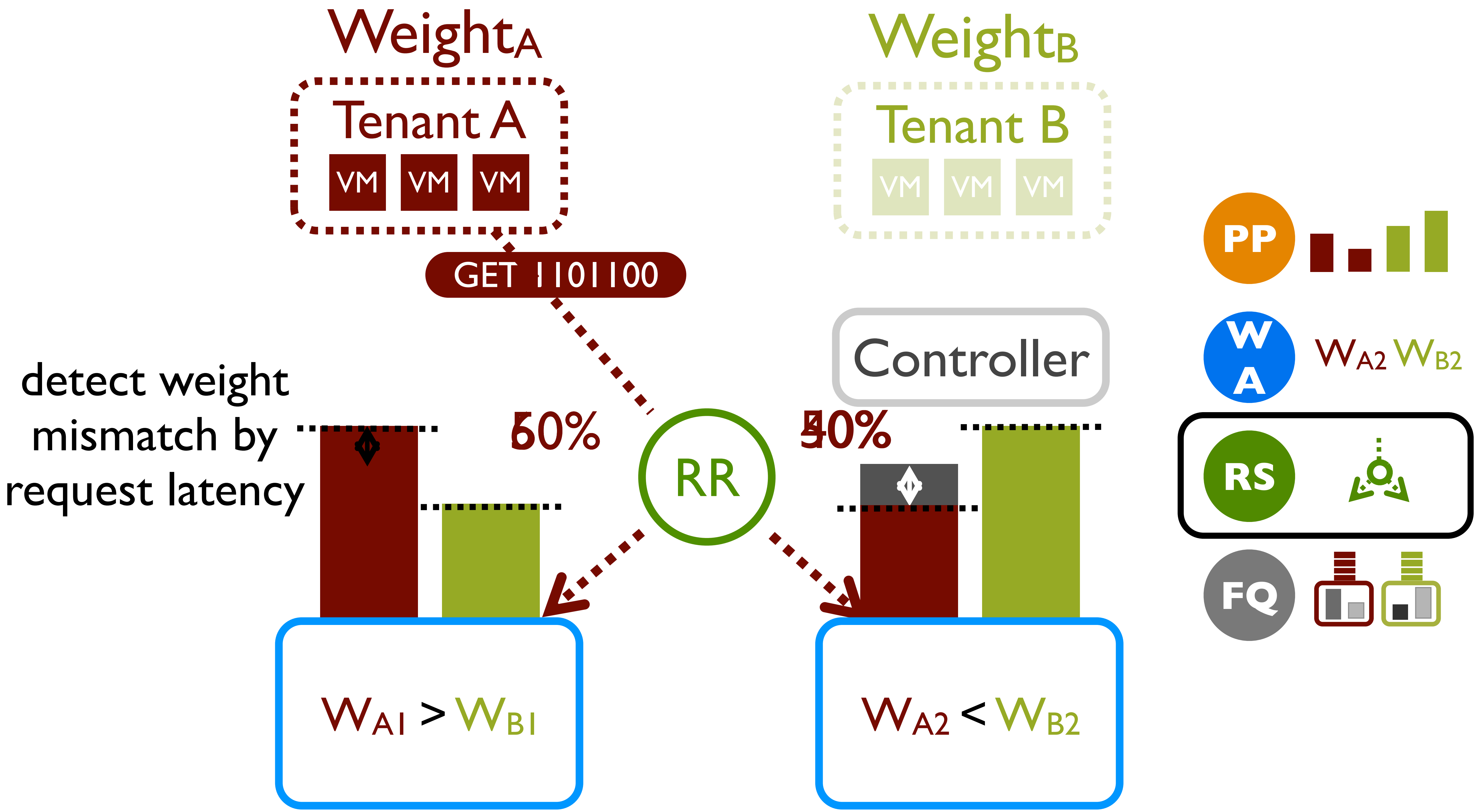
PP

W A W_{A2} W_{B2}

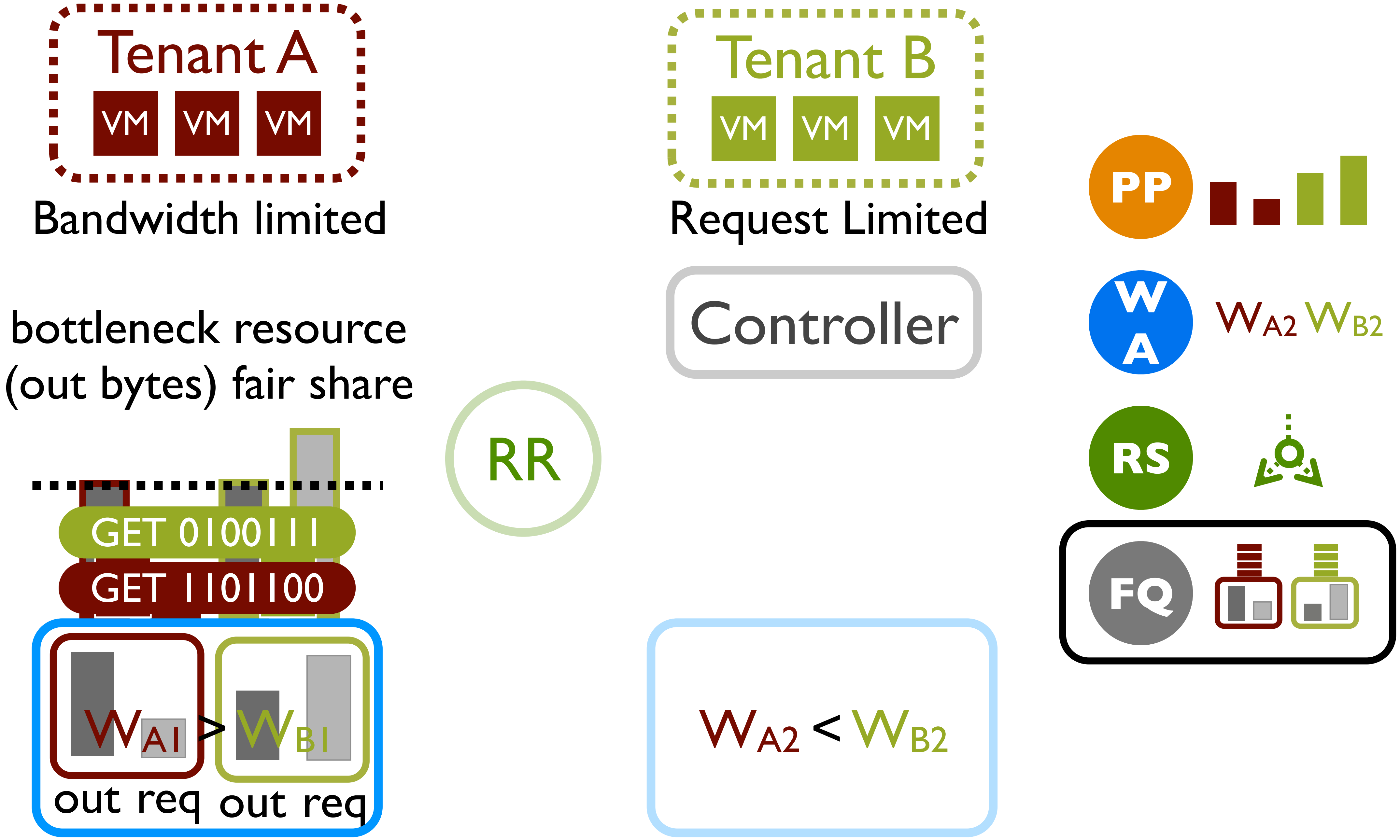
RS

FQ

Pisces: Select Replicas By Local Weight


detect weight mismatch by request latency

Strawman: Queue Tenants By Single Resource



Tenant A

VM VM VM

Bandwidth limited

bottleneck resource (out bytes) fair share

GET 0100111

GET 1101100

$W_{A1} > W_{B1}$

out req out req

RR

Tenant B

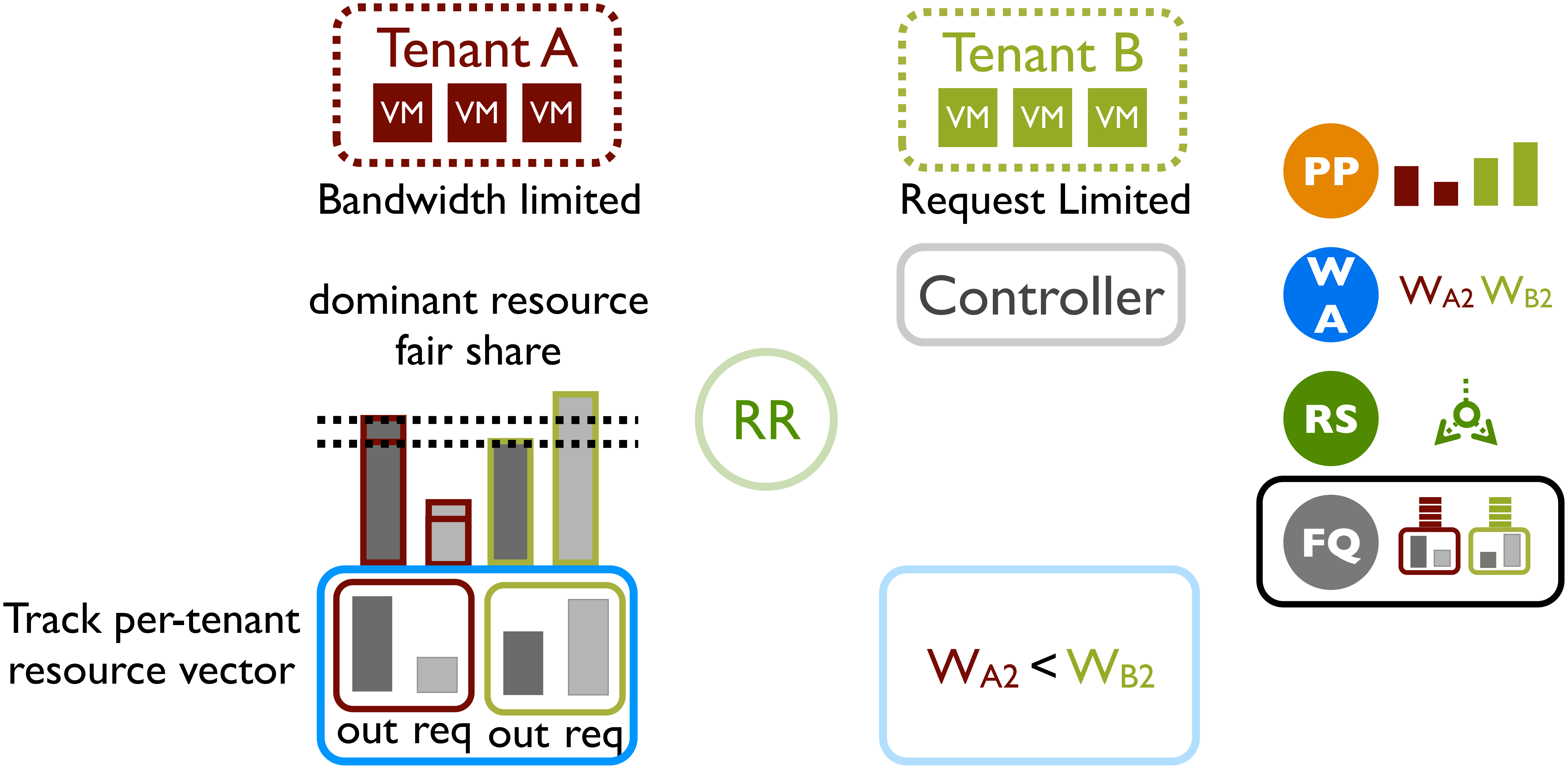
VM VM VM

Request Limited

Controller

$W_{A2} < W_{B2}$

Pisces: Queue Tenants By Dominant Resource



Tenant A

VM VM VM

Bandwidth limited

Tenant B

VM VM VM

Request Limited

Controller

dominant resource fair share

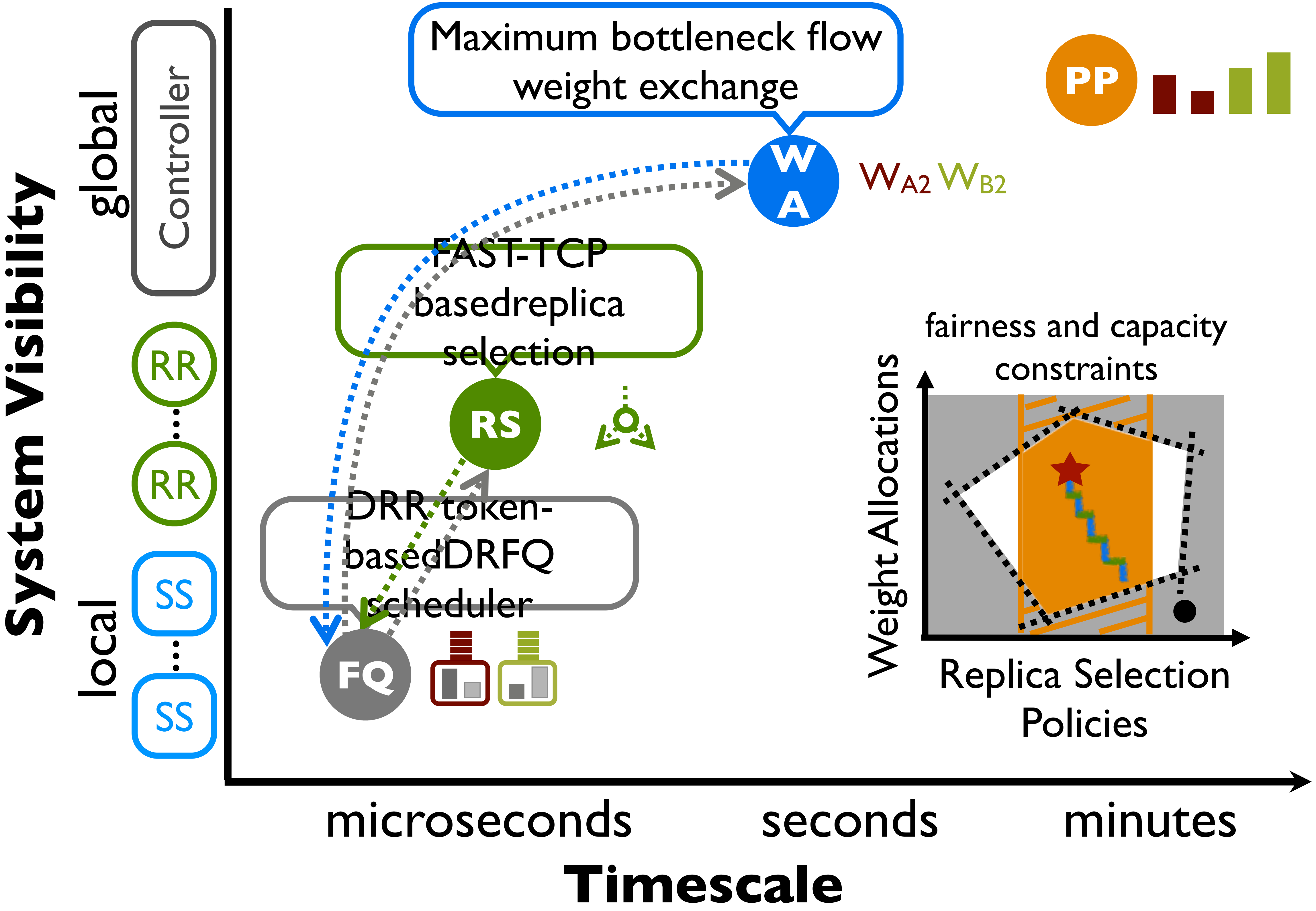
RR

Track per-tenant resource vector

out req out req

$W_{A2} < W_{B2}$

Pisces Mechanisms Solve For Global Fairness



Maximum bottleneck flow weight exchange

FAST-TCP basedreplica selection

DRR token-basedDRFQ scheduler

fairness and capacity constraints

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?

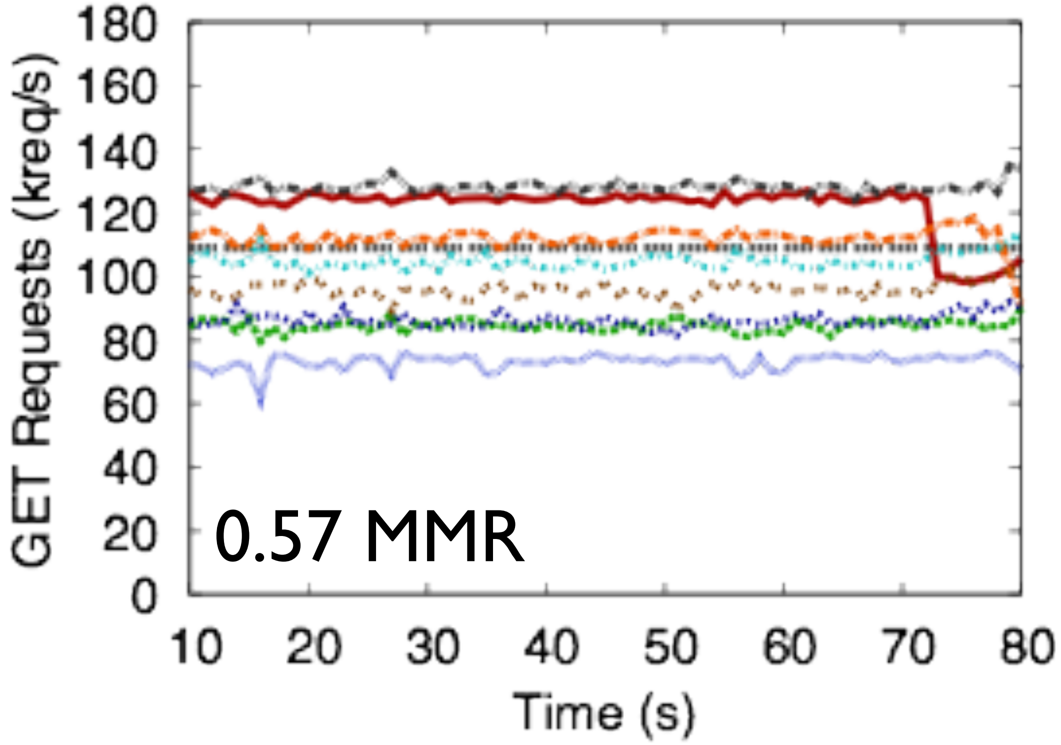
Evaluation

- 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

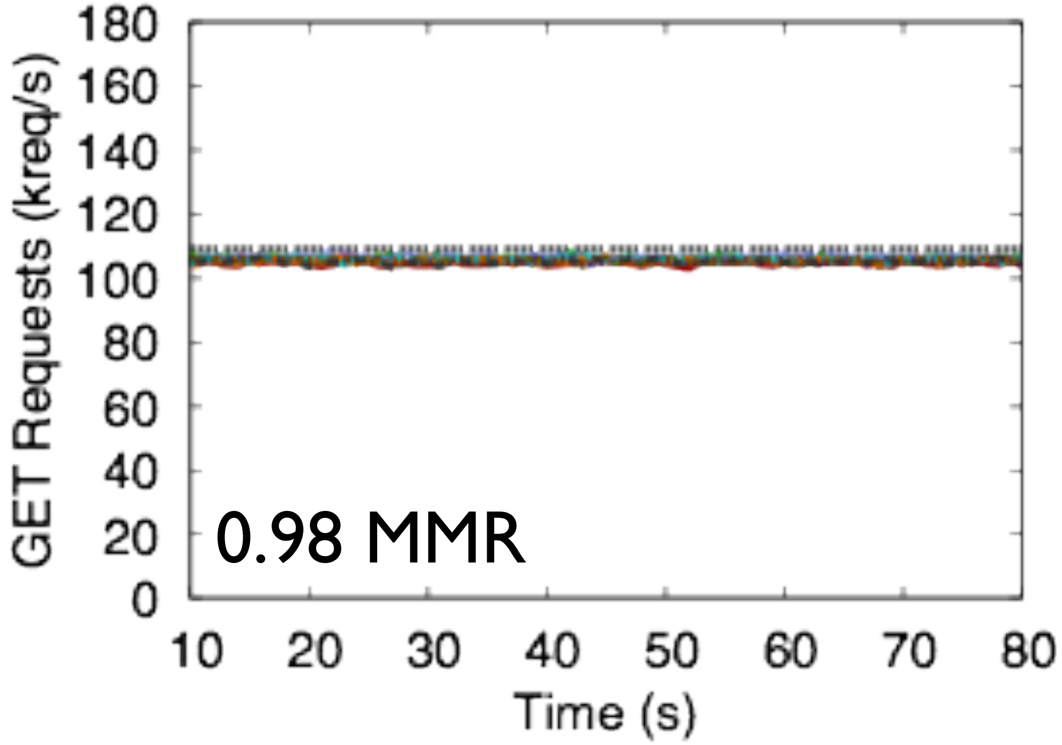
Ideal fair share: 110 kreq/s (1kB requests)

Unmodified Membase



0.57 MMR

Pisces



0.98 MMR

8 Tenants - 8 Client - 8 Storage Nodes

Zipfian object popularity distribution

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

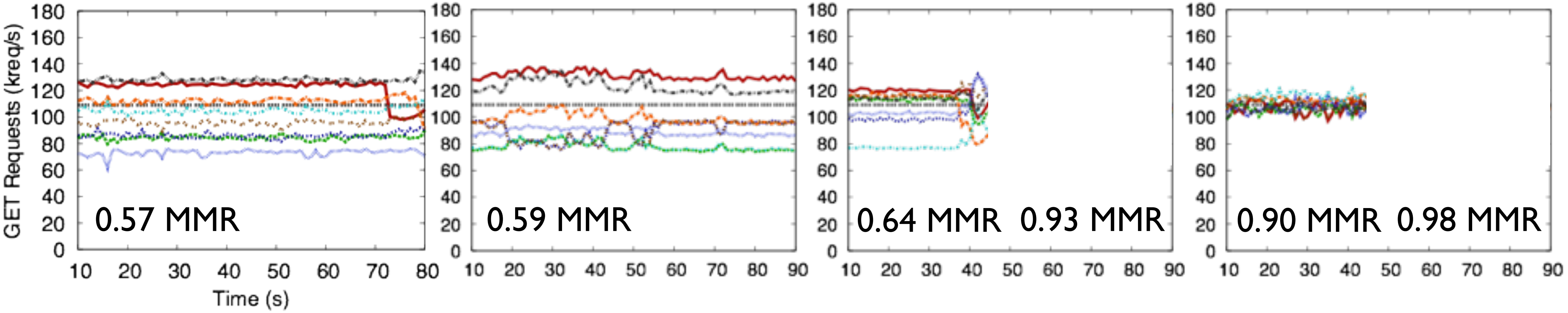
Each Pisces Mechanism Contributes to System-wide Fairness and Isolation

Unmodified Membase

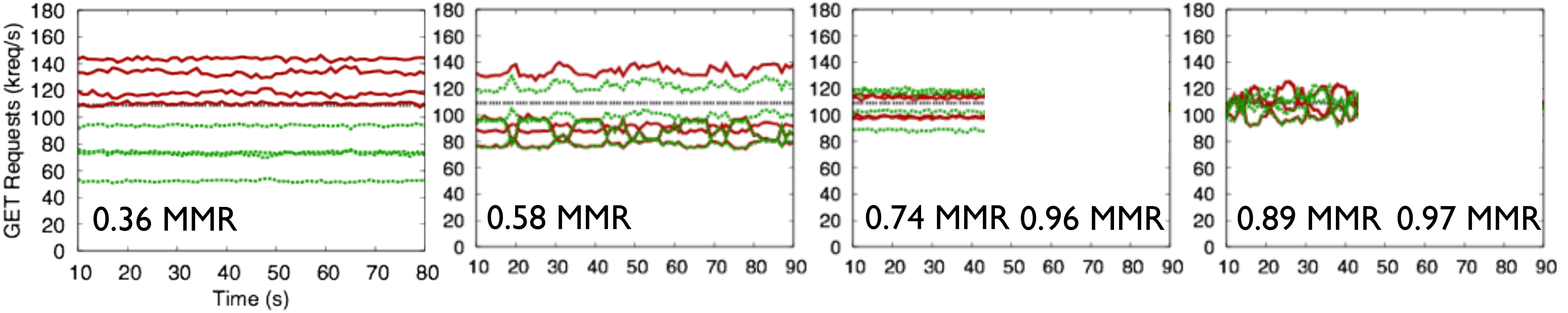
FQ

FQ PP WA

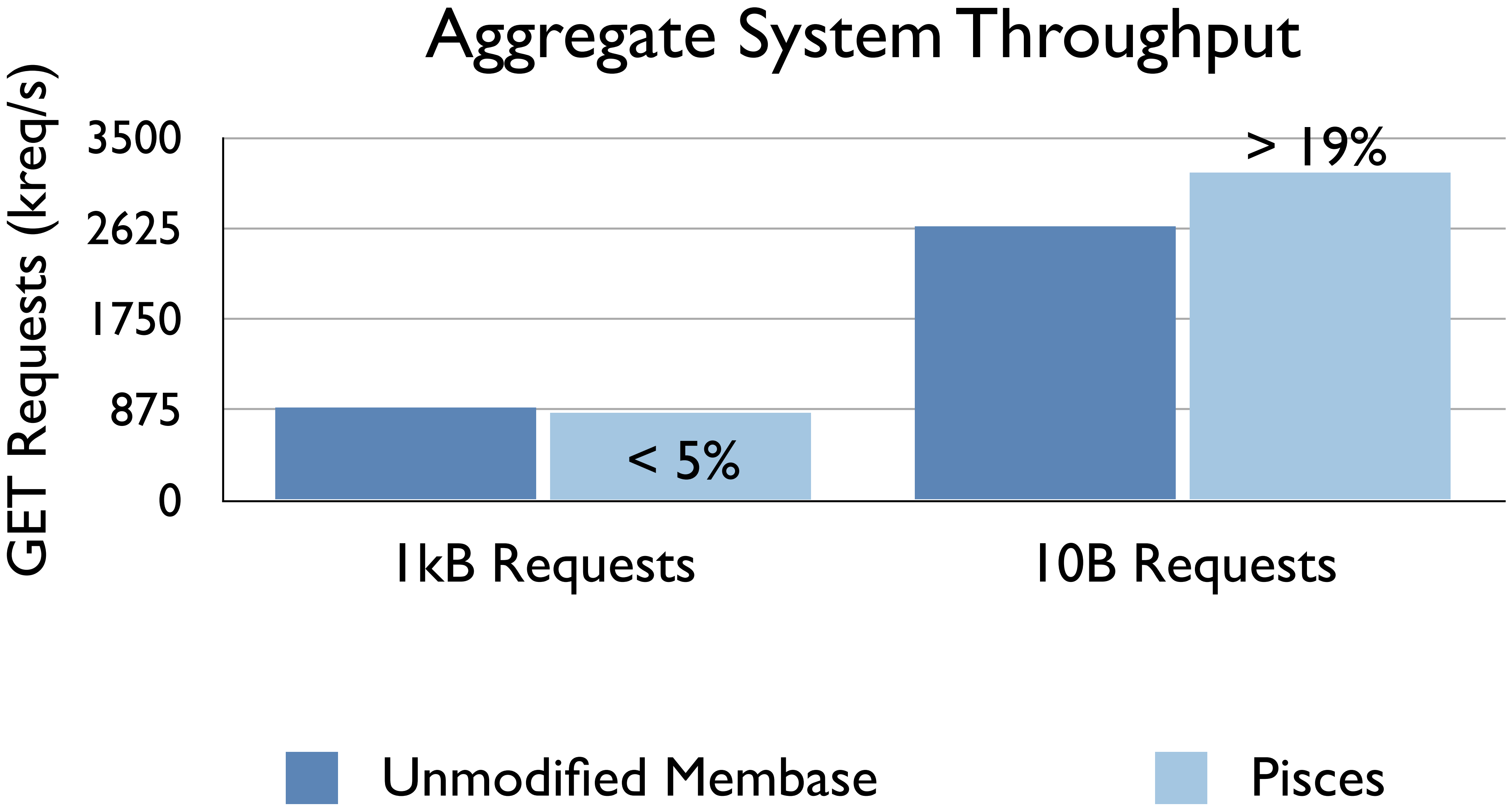
FQ PP RS WA



2x vs 1x demand



Pisces Imposes Low-overhead

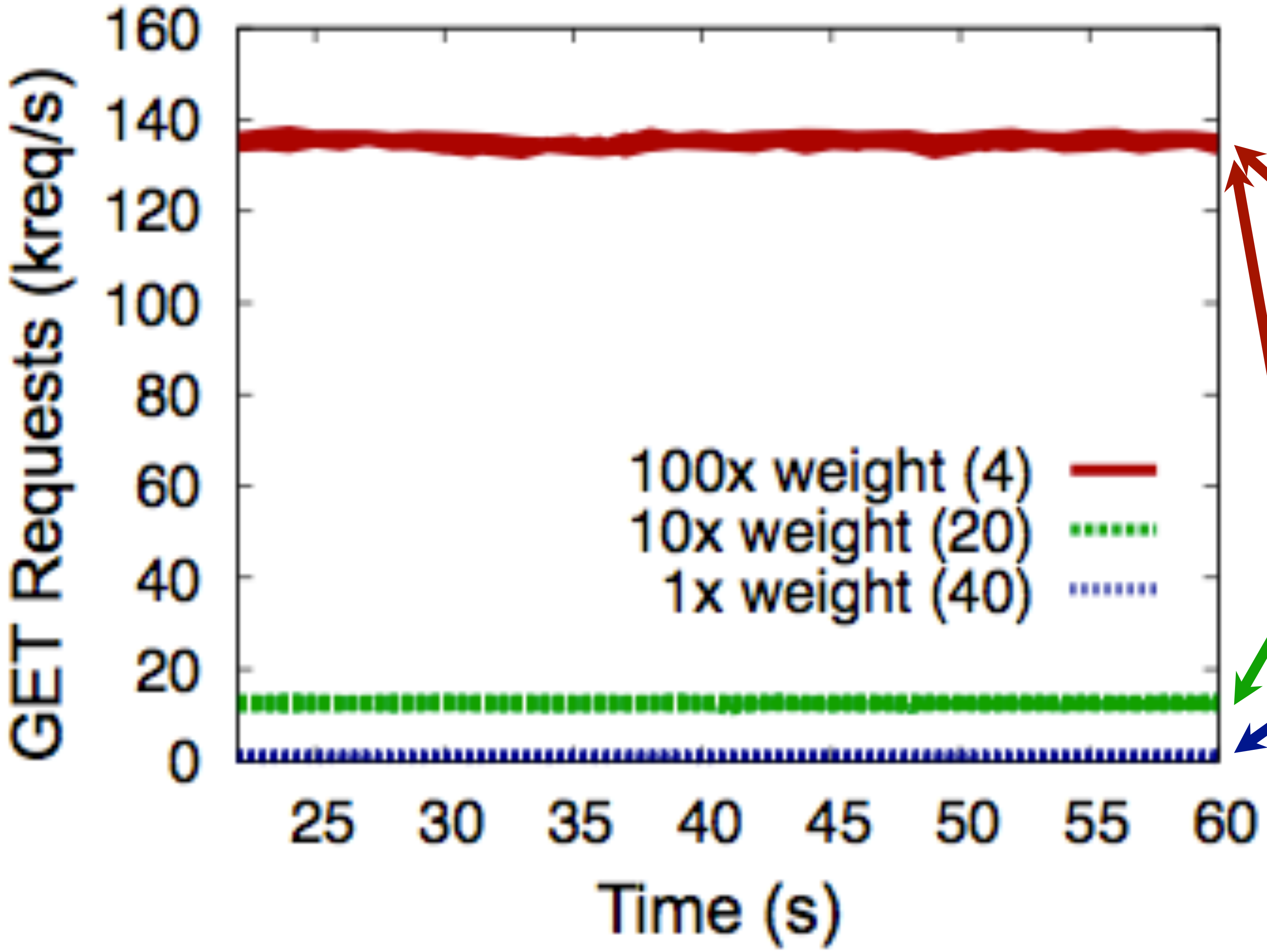


Pisces Achieves System-wide Weighted Fairness

0.98 MMR
4 heavy hitters

0.89 MMR
20 moderate demand

0.91 MMR
40 low demand

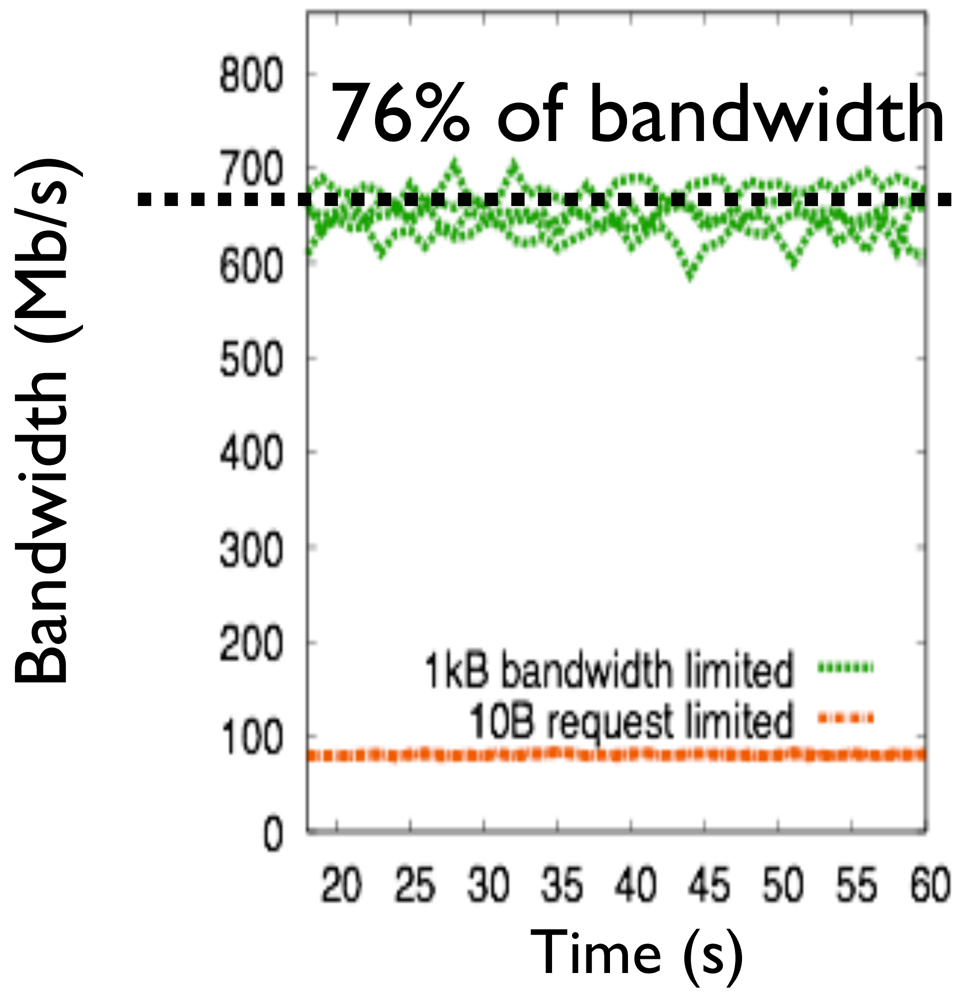


0.91 MMR

0.56 MMR

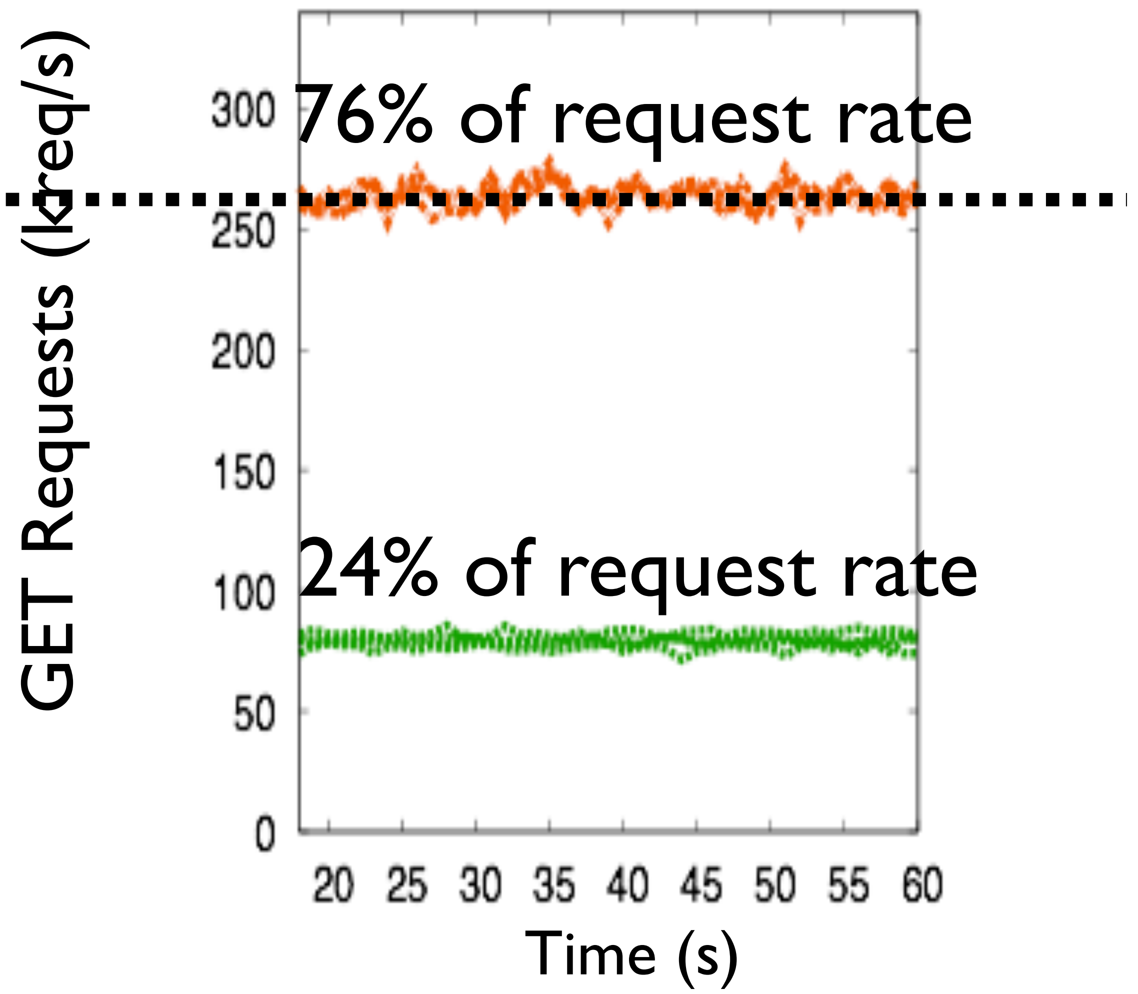
Pisces Achieves Dominant Resource Fairness

1kB workload
bandwidth limited



76% of bandwidth

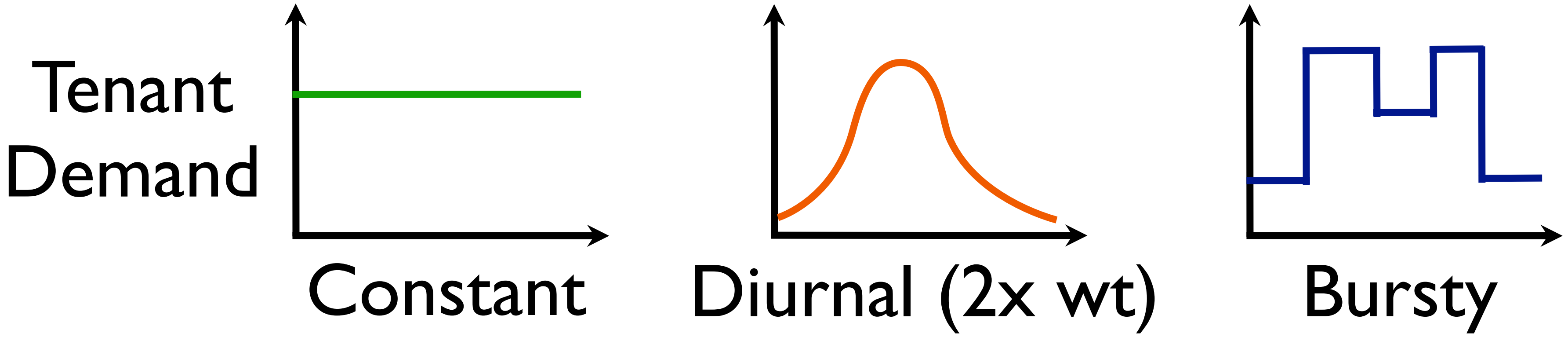
10B workload
request limited

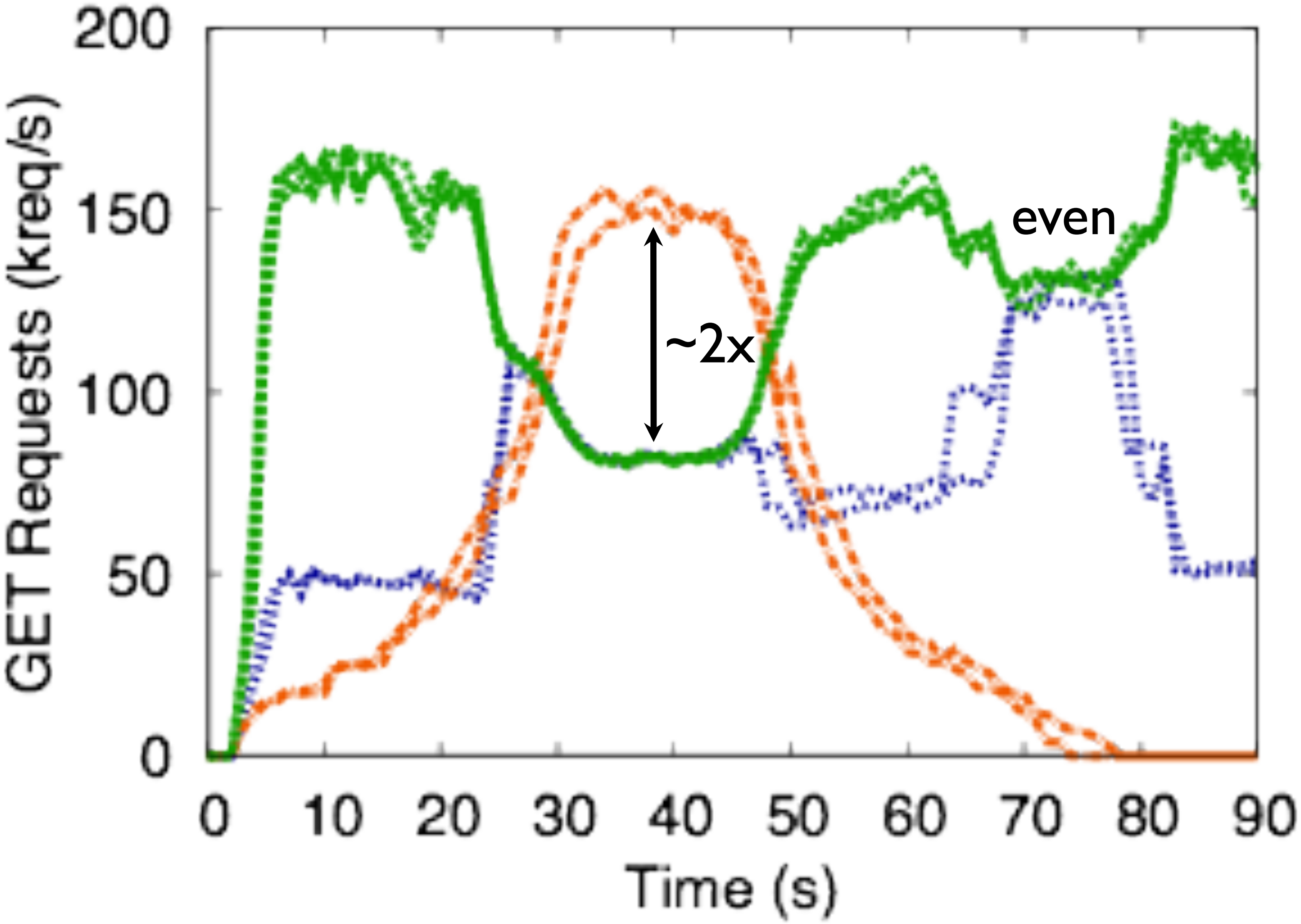


76% of request rate

24% of request rate

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

Thank you

Performance Isolation and Fairness for Multi-Tenant Cloud Storage

PISCES

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

*Princeton ✦IBM Research