PISCES

PrincetonUniversity

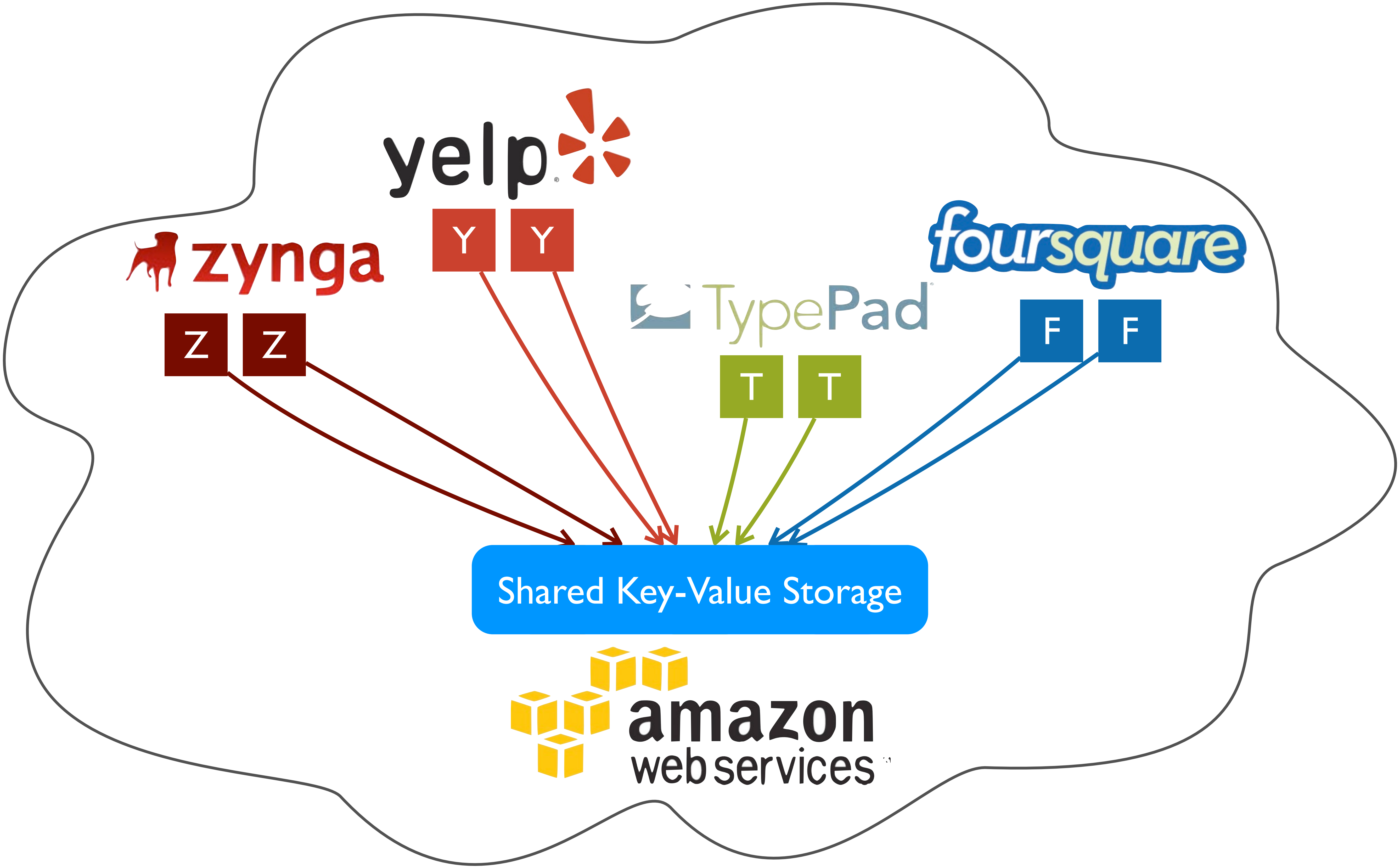
Performance Isolation and Fairness for Multi-Tenant Cloud Storage

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

sns.cs.princeton.edu

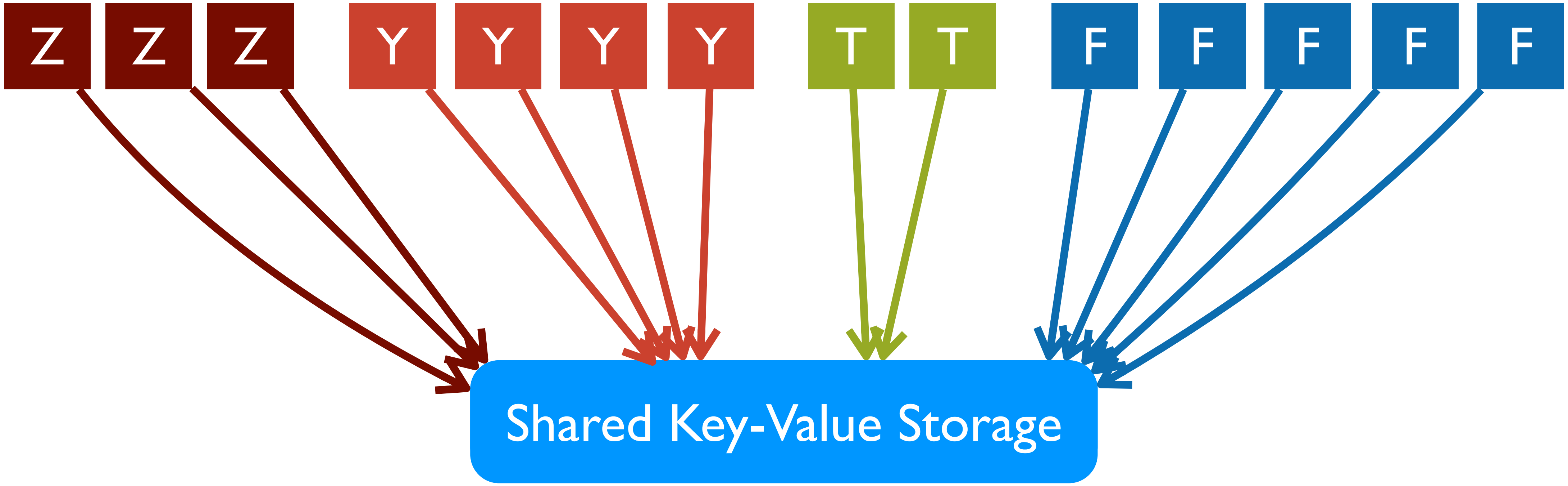
*Princeton ✦IBM Research

Setting: Shared Storage in the Cloud



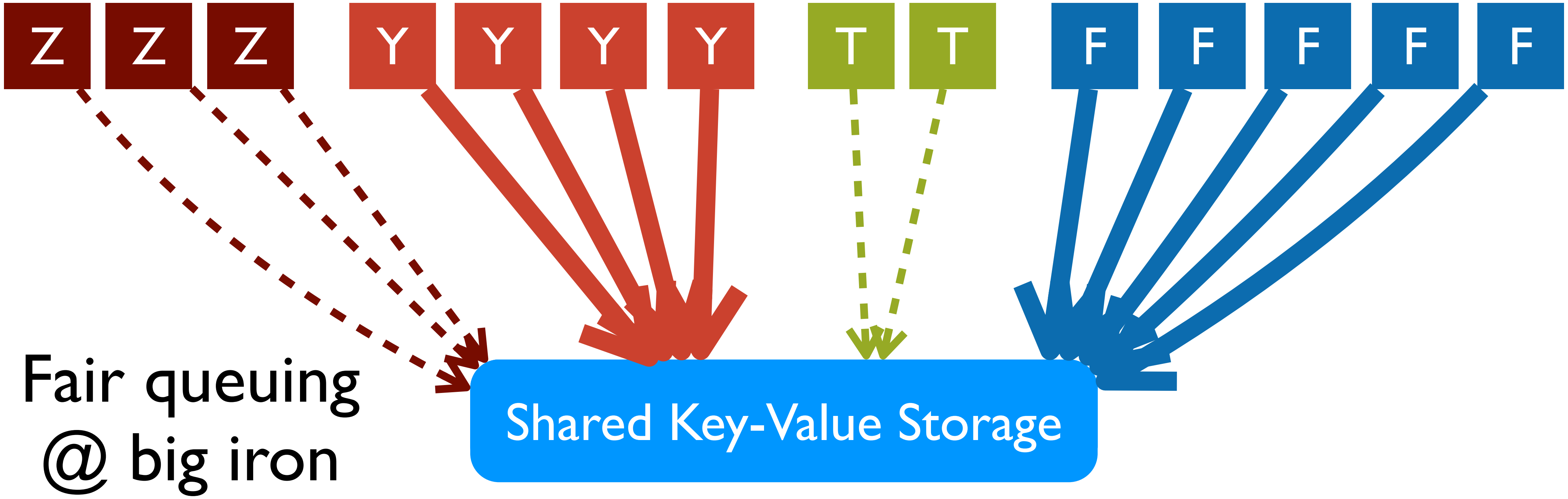
Shared Key-Value Storage

Predictable Performance is Hard



Multiple co-located tenants ⇒ resource contention

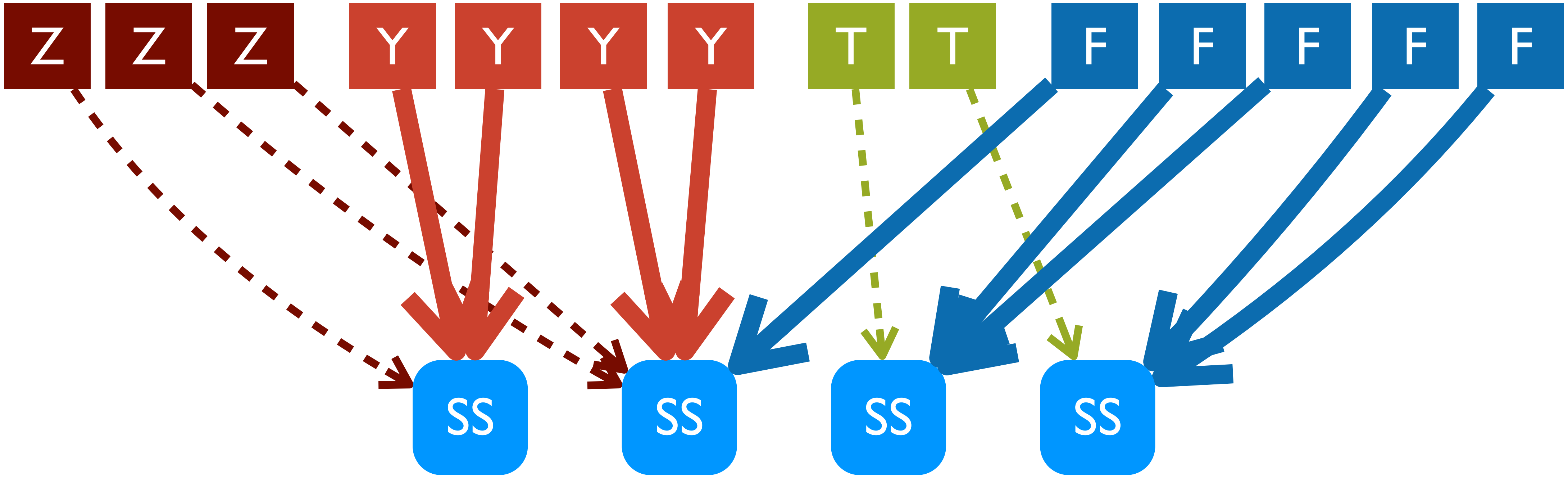
Predictable Performance is Hard



Fair queuing @ big iron

Multiple co-located tenants ⇒ resource contention

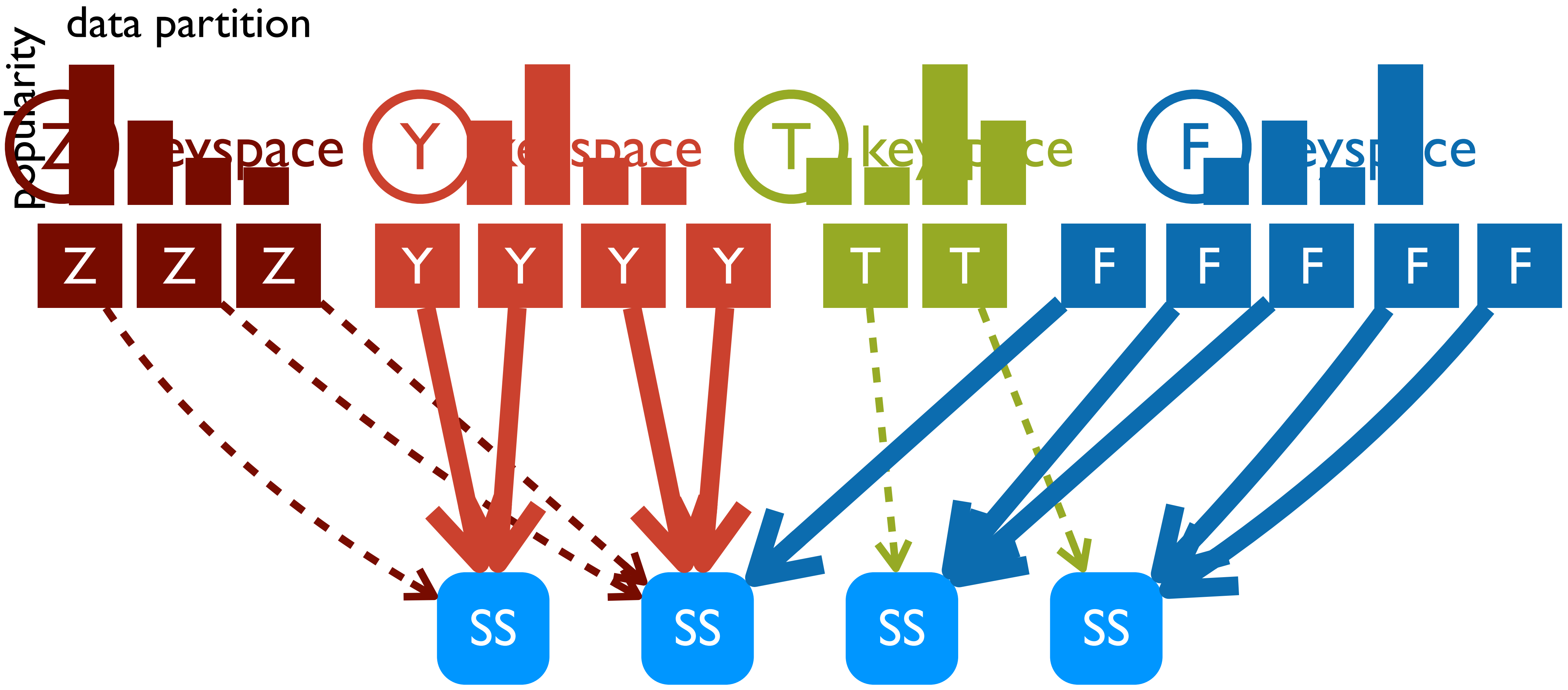
Predictable Performance is Hard



Multiple co-located tenants ⇒ resource contention

Distributed system ⇒ distributed resource allocation

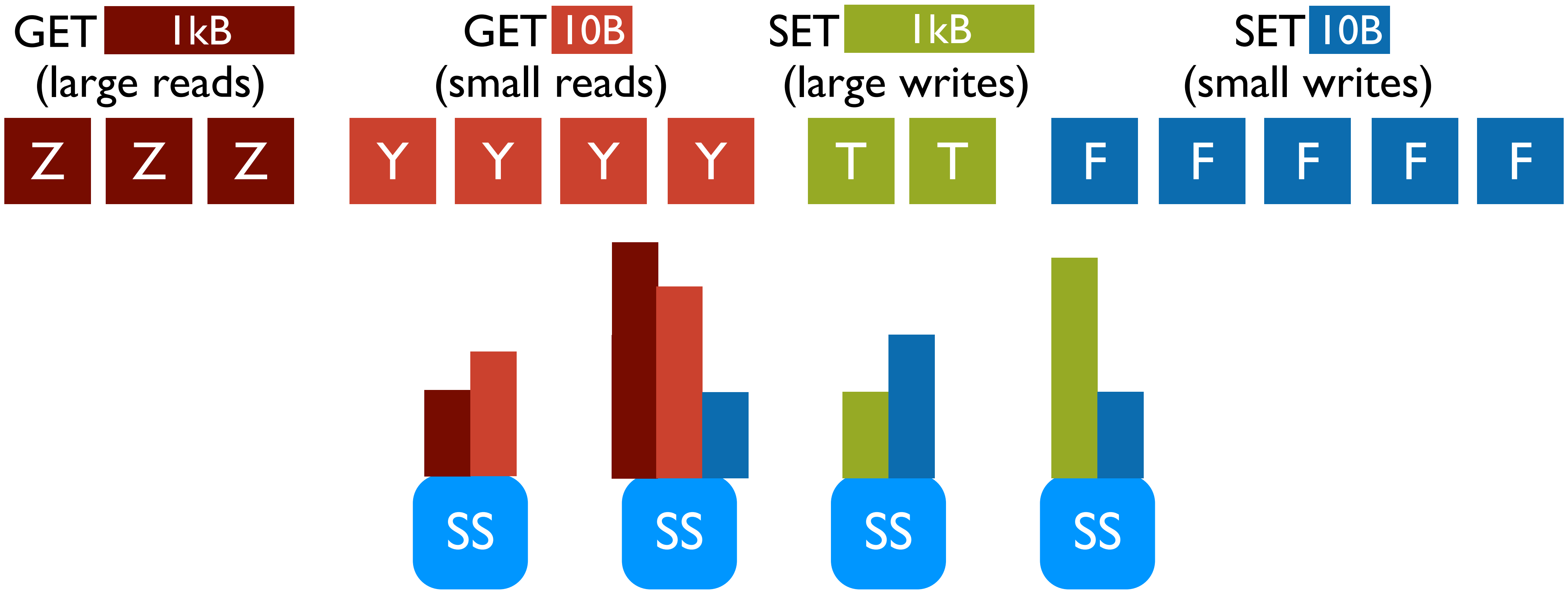
Predictable Performance is Hard



Multiple co-located tenants ⇒ resource contention

Distributed system ⇒ distributed resource allocation

Predictable Performance is Hard



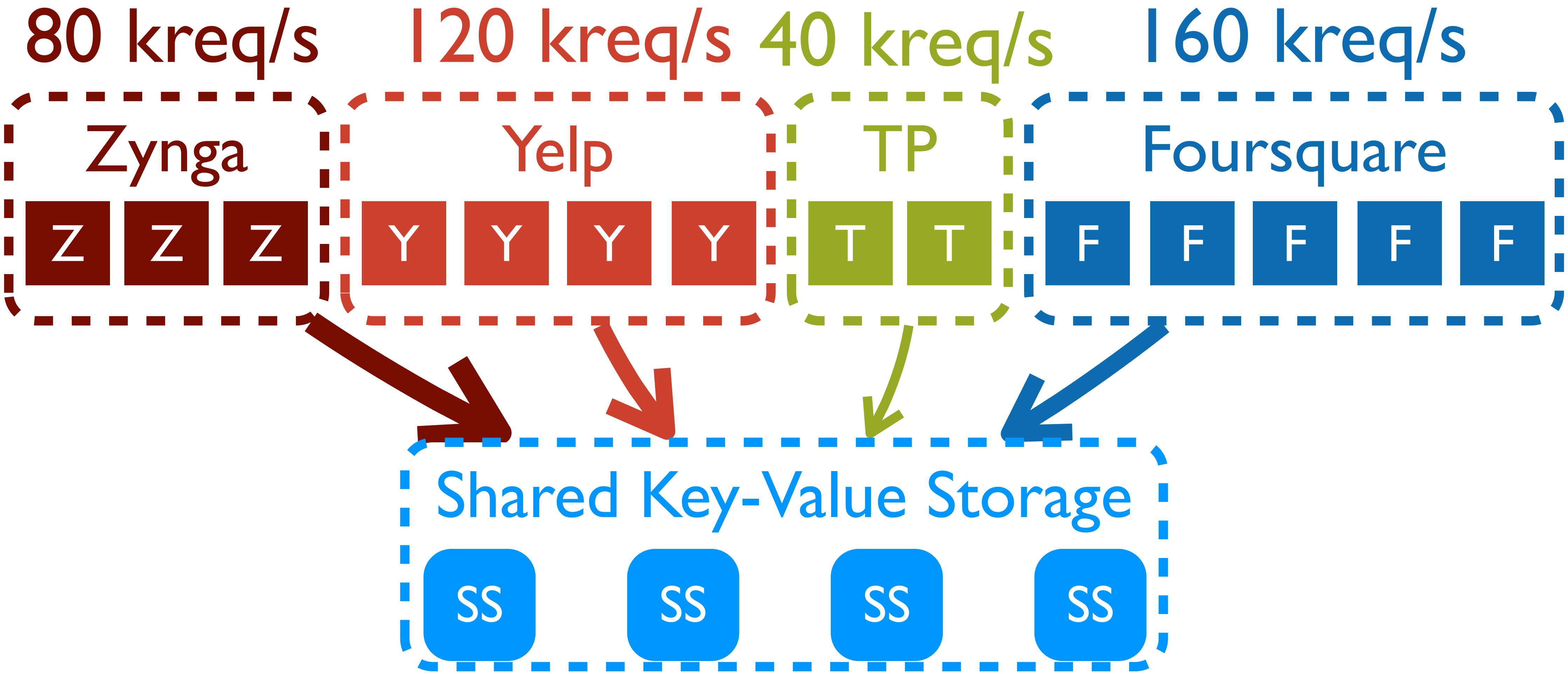
Multiple co-located tenants ⇒ resource contention

Distributed system ⇒ distributed resource allocation

Skewed object popularity ⇒ variable per-node demand

Disparate workloads ⇒ different bottleneck resources

Tenants Want System-wide Resource Guarantees



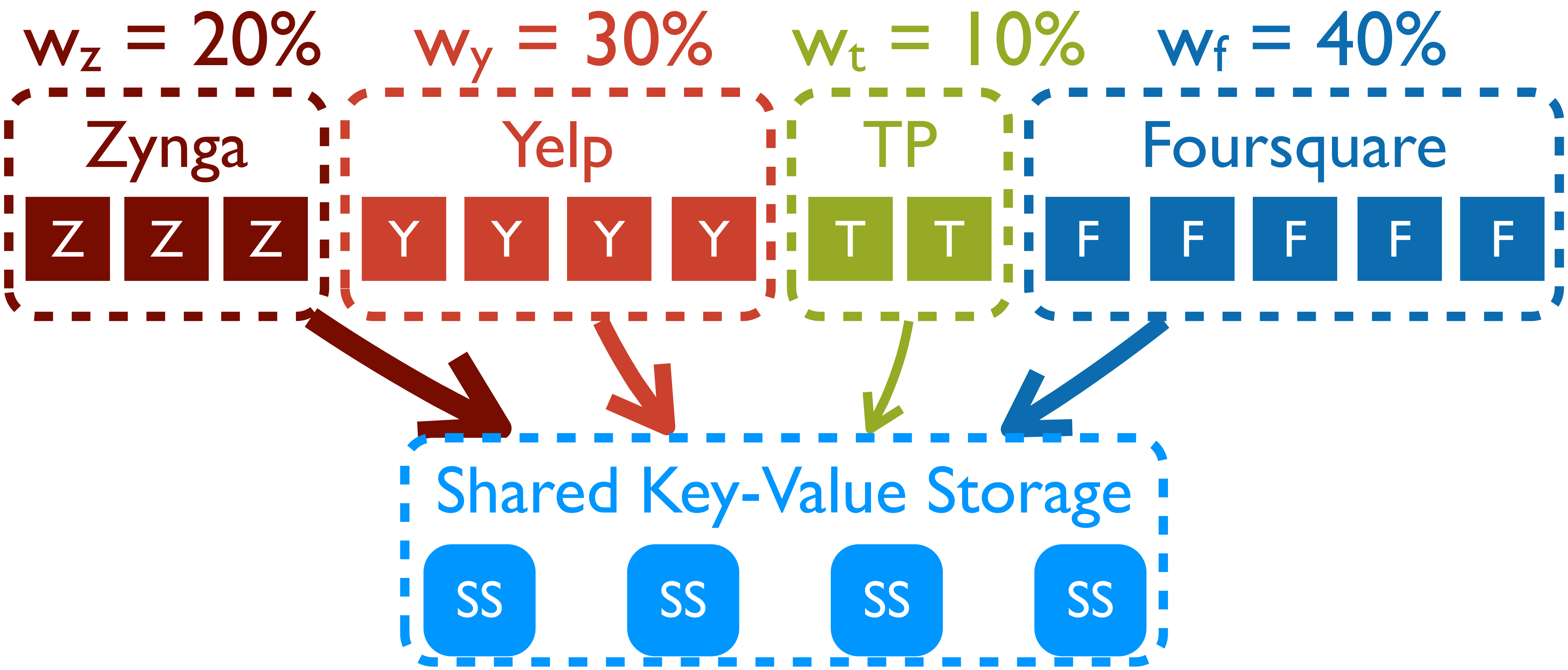
Multiple co-located tenants ⇒ resource contention

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

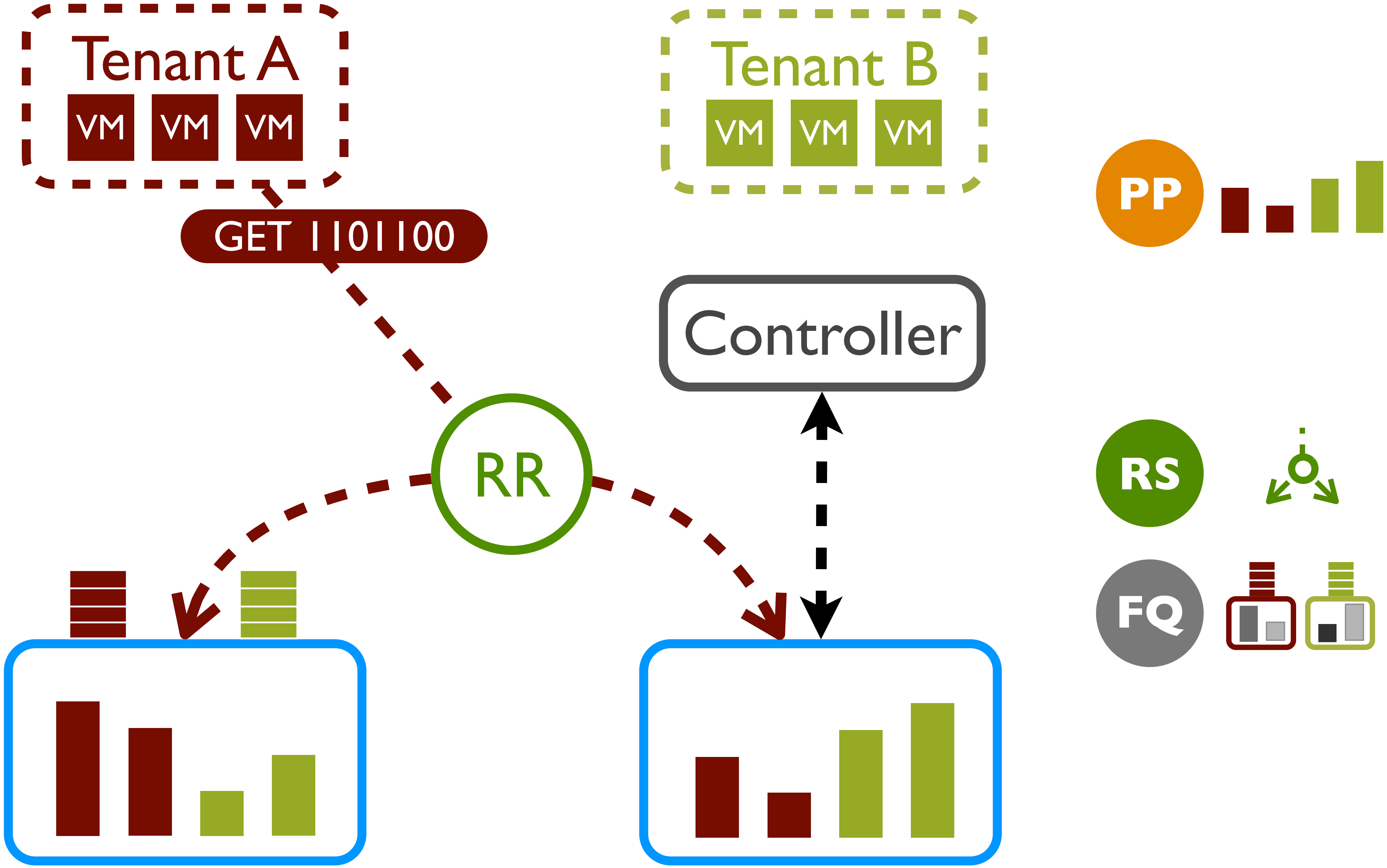
Pisces: Predictable Shared Cloud Storage

- Pisces
  - Per-tenant max-min fair shares of system-wide resources ~ min guarantees, high utilization
  - Arbitrary object popularity
  - Different resource bottlenecks
- Pisces Target Environment
  - Basic Key-Value Store ~ GET/SET
  - Asynchronous durability ~ MyISAM, Membase
  - Well-provisioned network ~ full bisectional bandwidth
  - Moderate object popularity shift ~ order of minutes

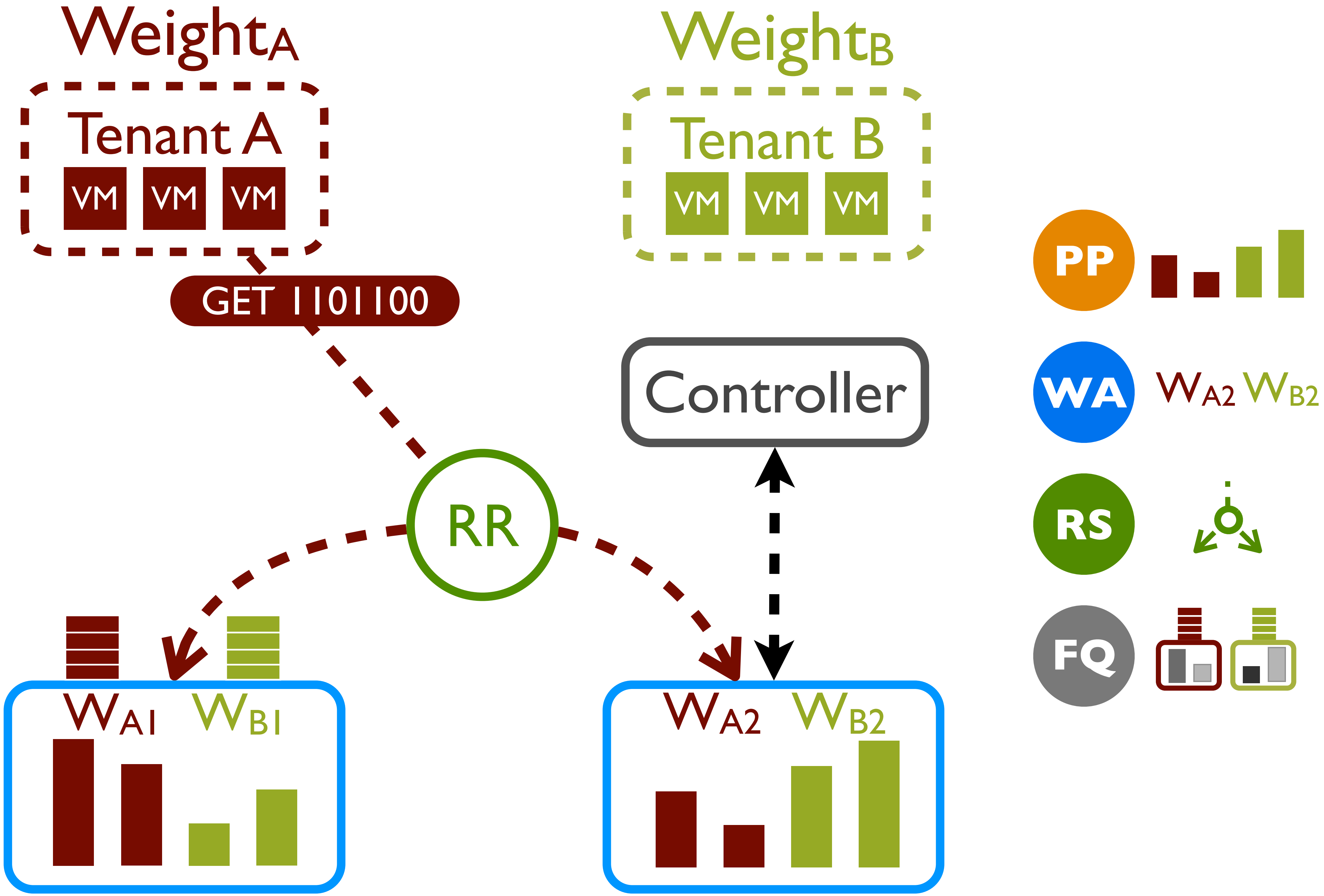
Pisces: Predictable Shared Cloud Storage

- 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)

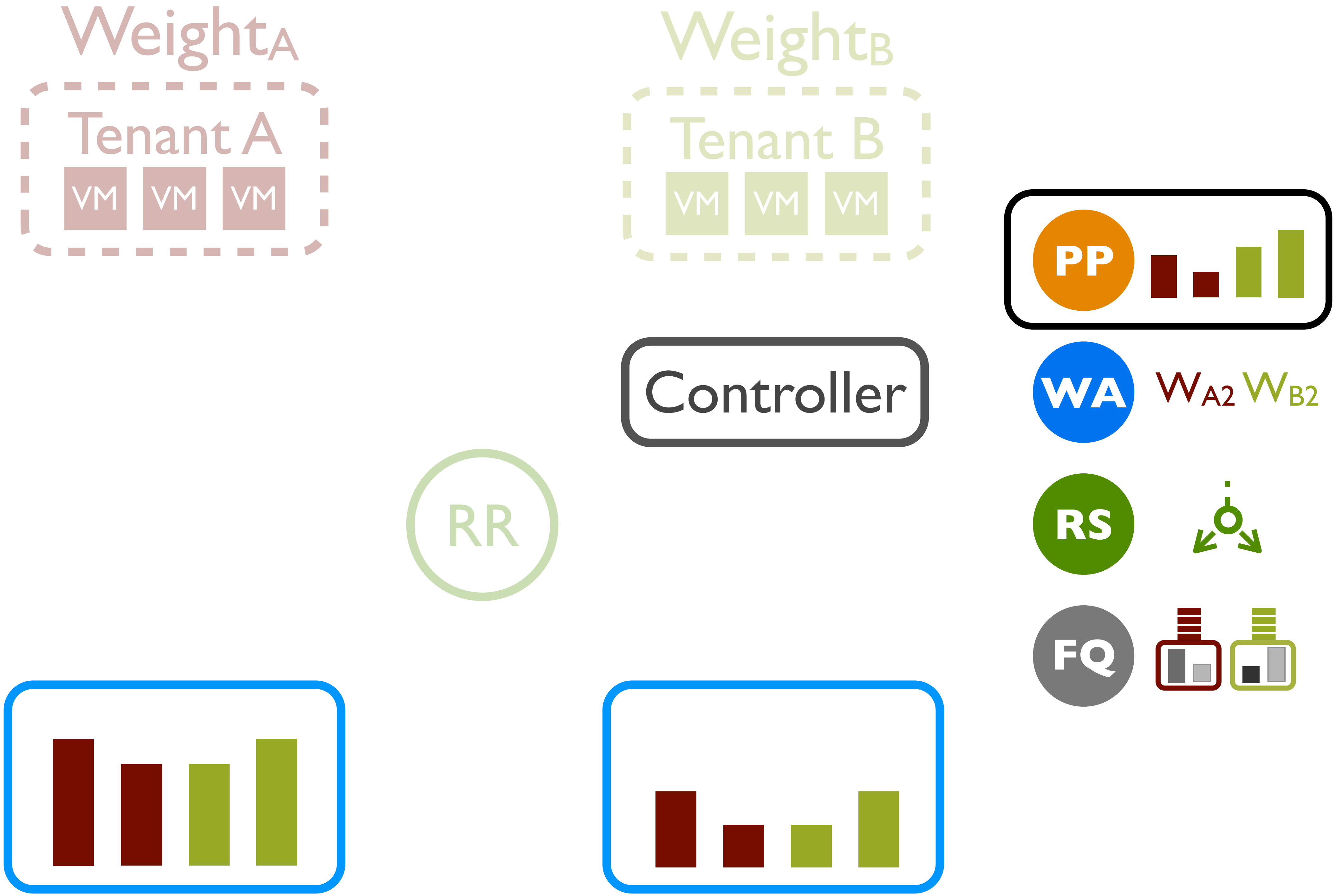
Predictable Multi-Tenant Key-Value Storage



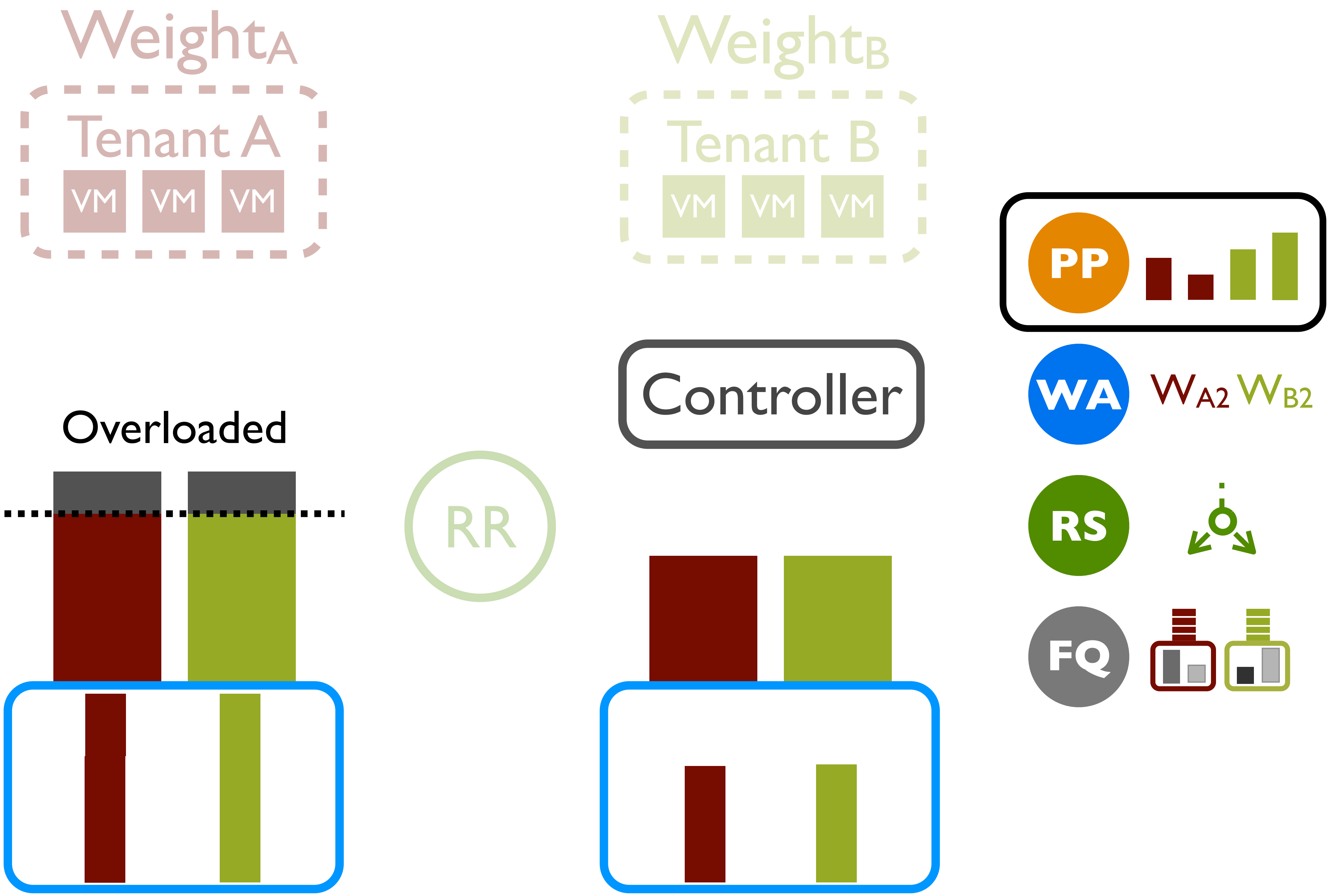
Predictable Multi-Tenant Key-Value Storage



Strawman: Place Partitions Randomly



Strawman: Place Partitions Randomly



Weight$_A$

Tenant A

VM VM VM

Weight$_B$

Tenant B

VM VM VM

PP

Controller

WA W_{A2} W_{B2}

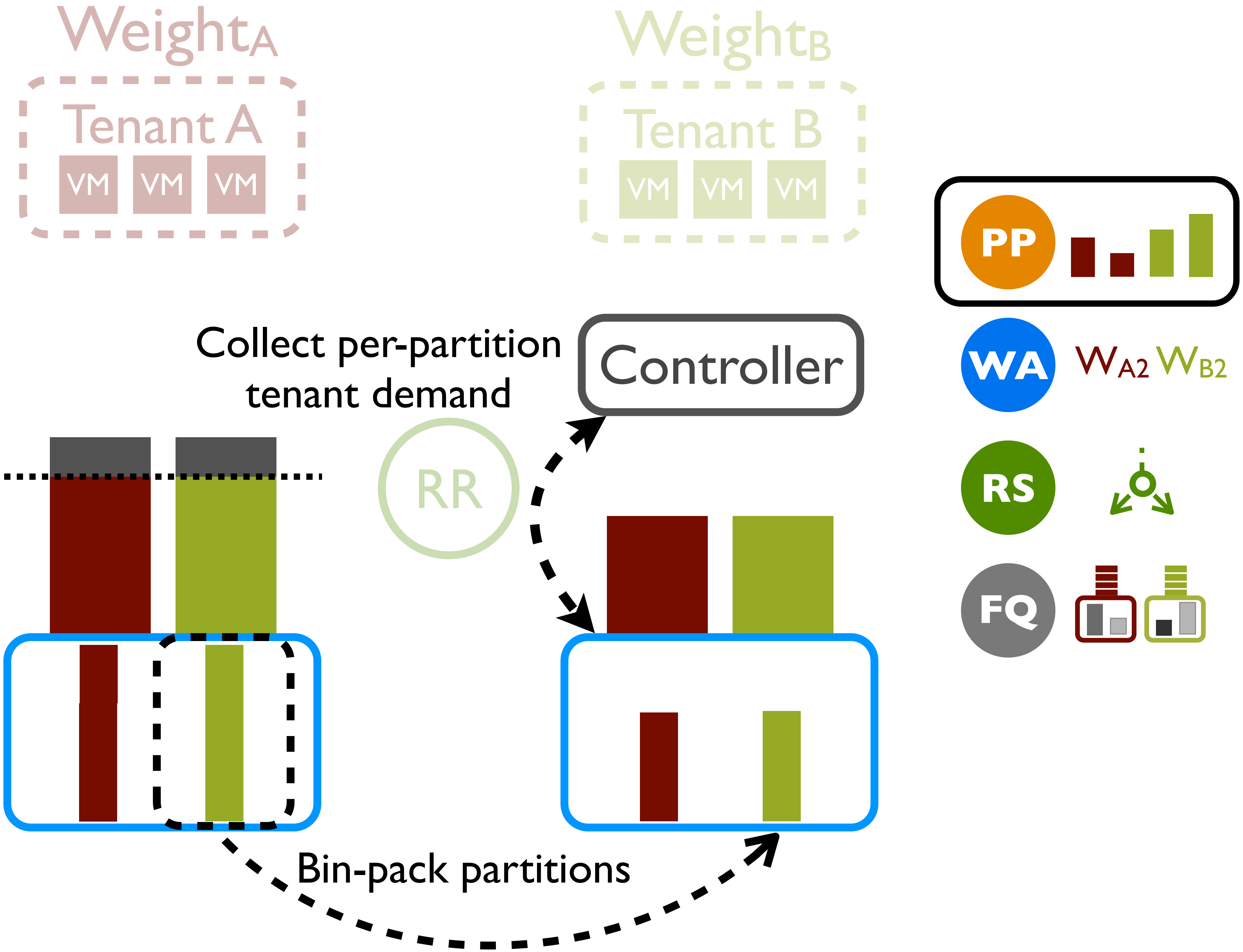
Overloaded

RR

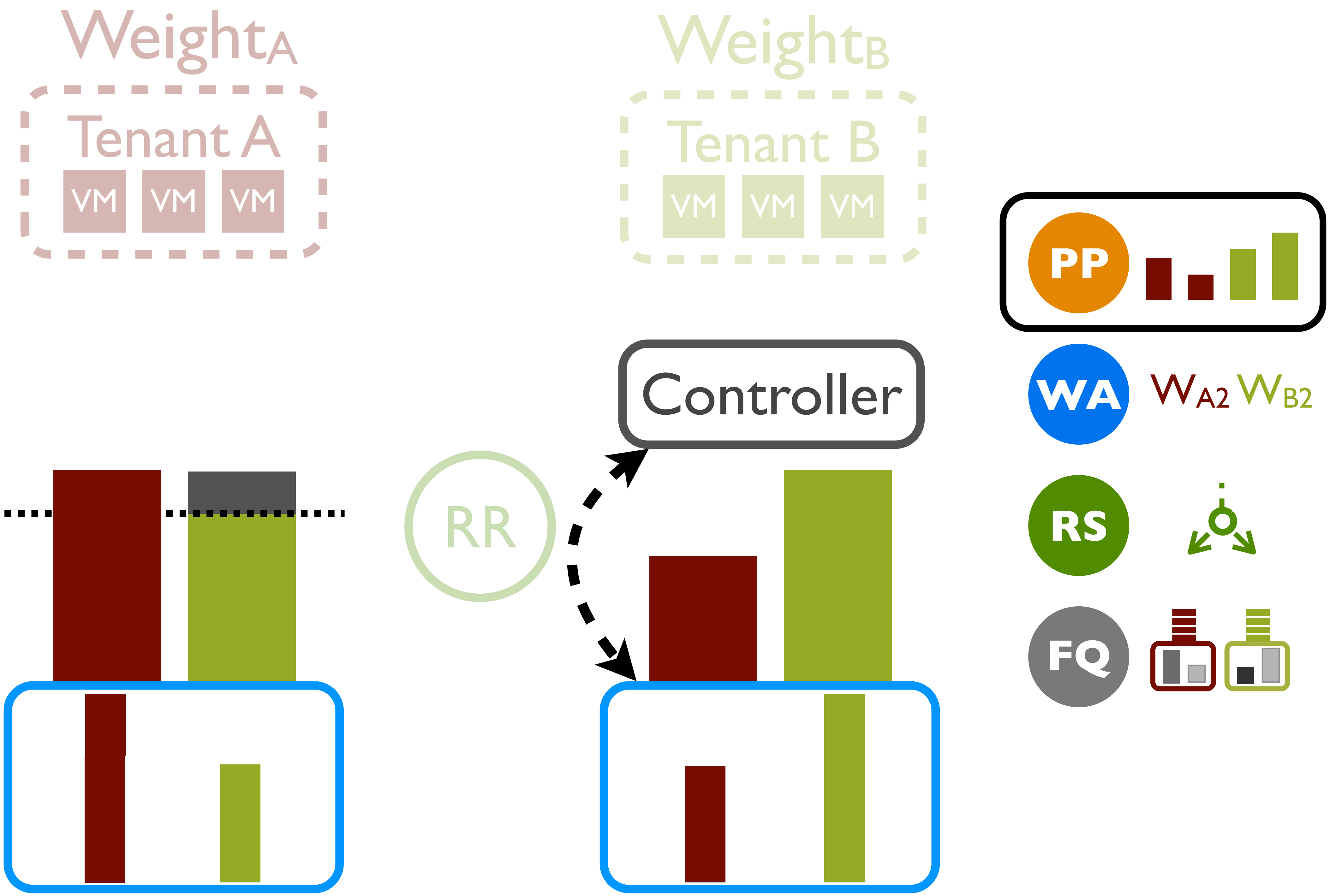
RS

FQ

Pisces: Place Partitions By Fairness Constraints

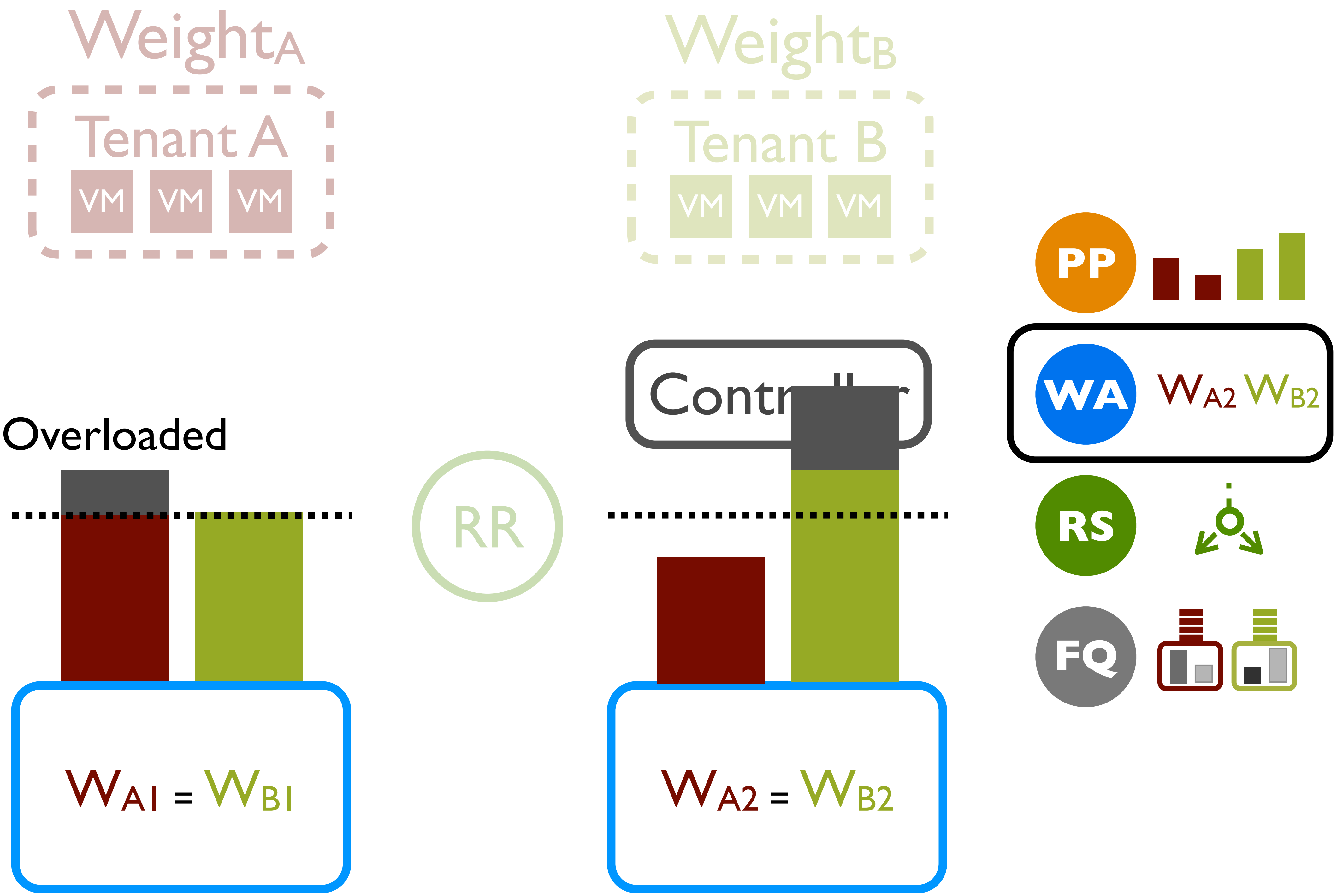


Pisces: Place Partitions By Fairness Constraints

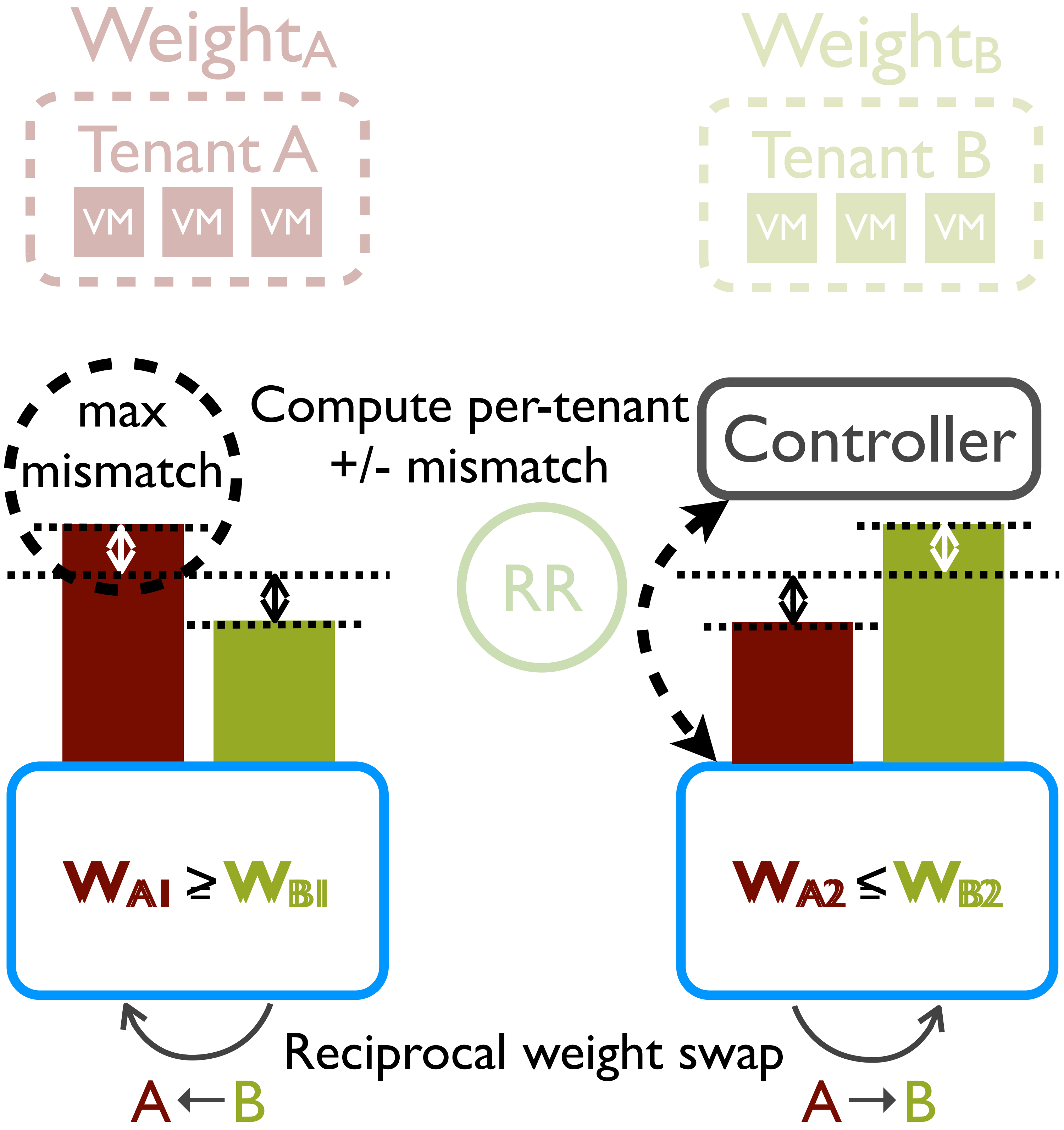


Results in feasible partition placement

Strawman: Allocate Local Weights Evenly



Pisces: Allocate Local Weights By Tenant Demand

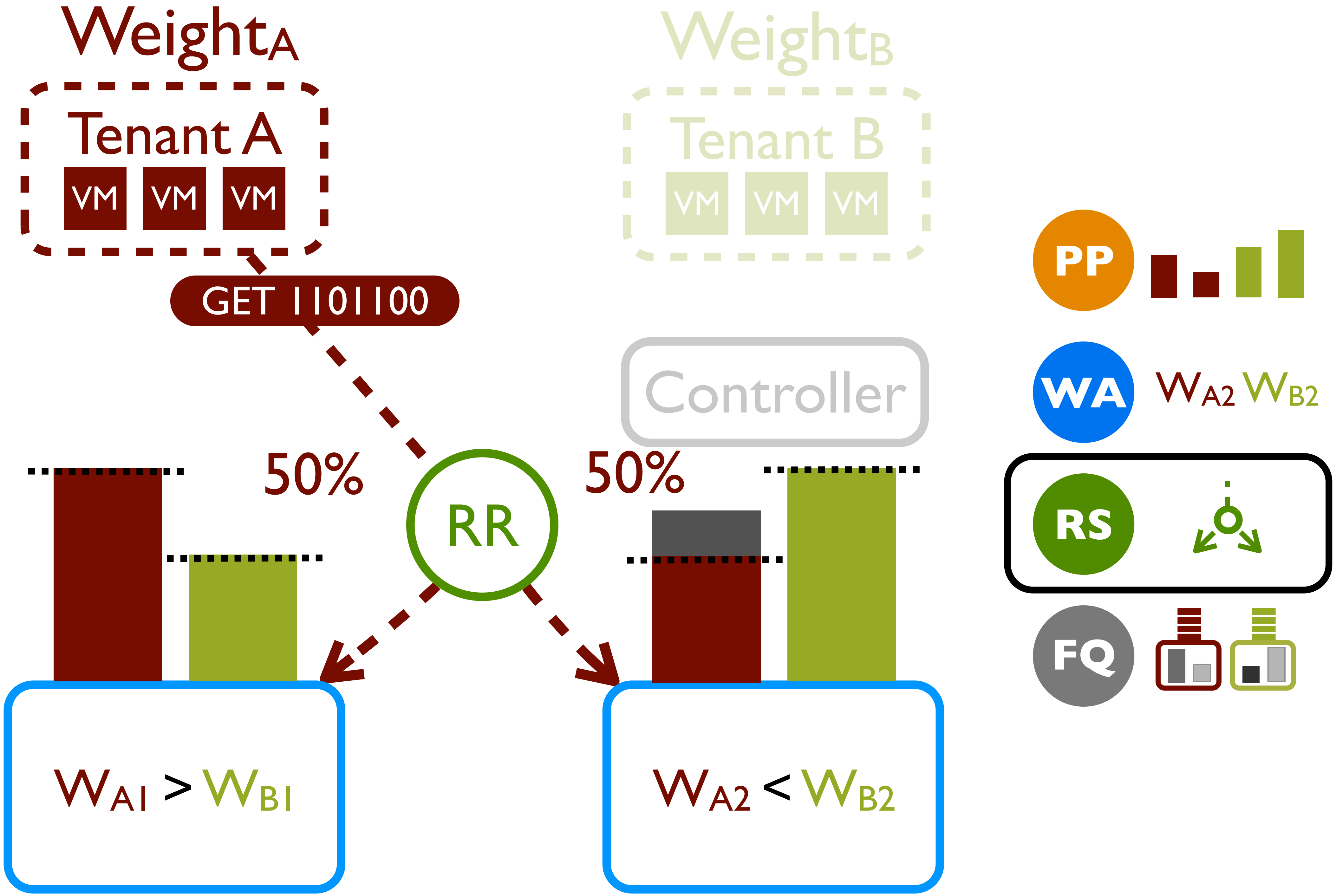


Compute per-tenant +/- mismatch

Reciprocal weight swap

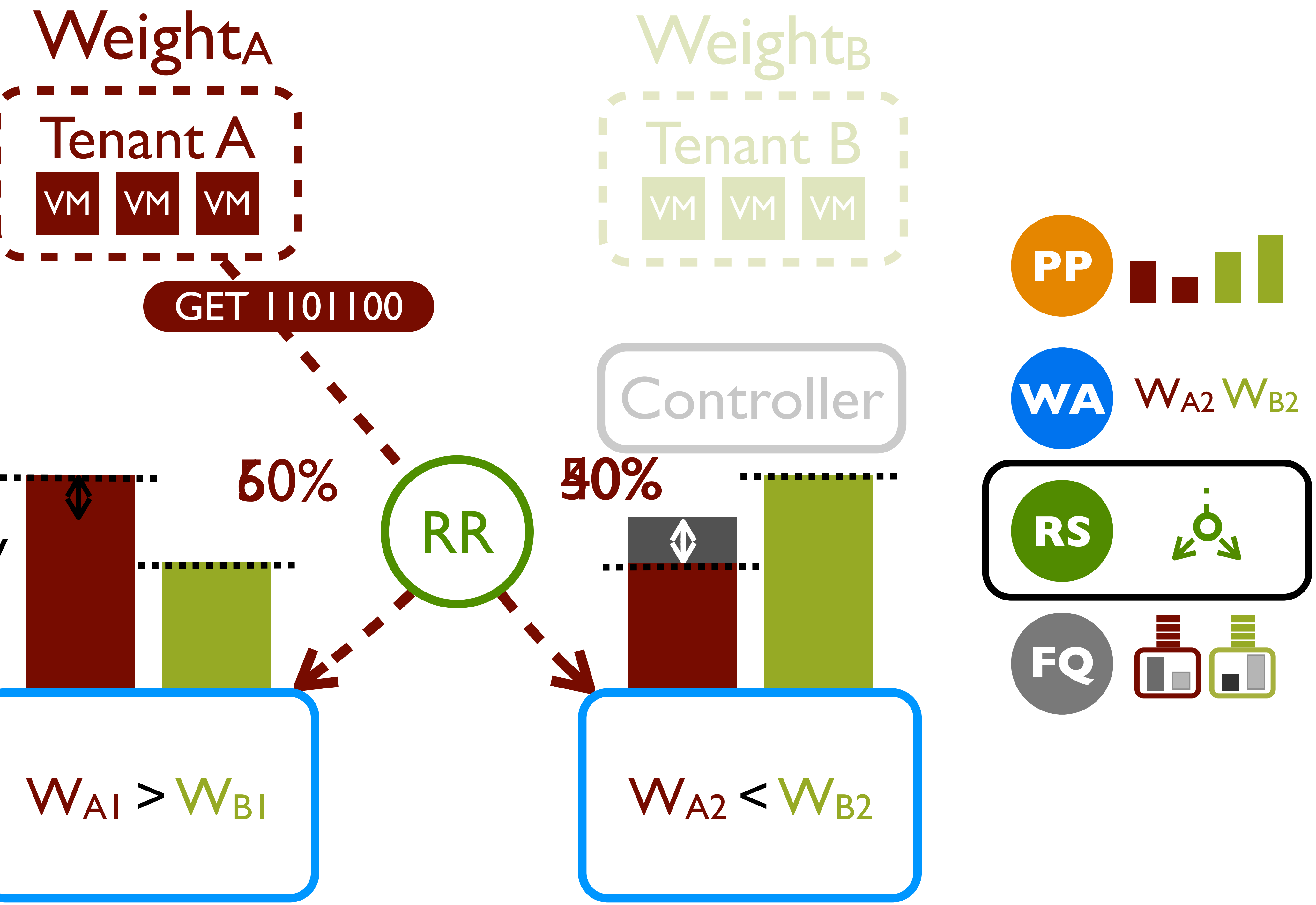
PP
WA
W_{A2} W_{B2}
RS
FQ

Strawman: Select Replicas Evenly

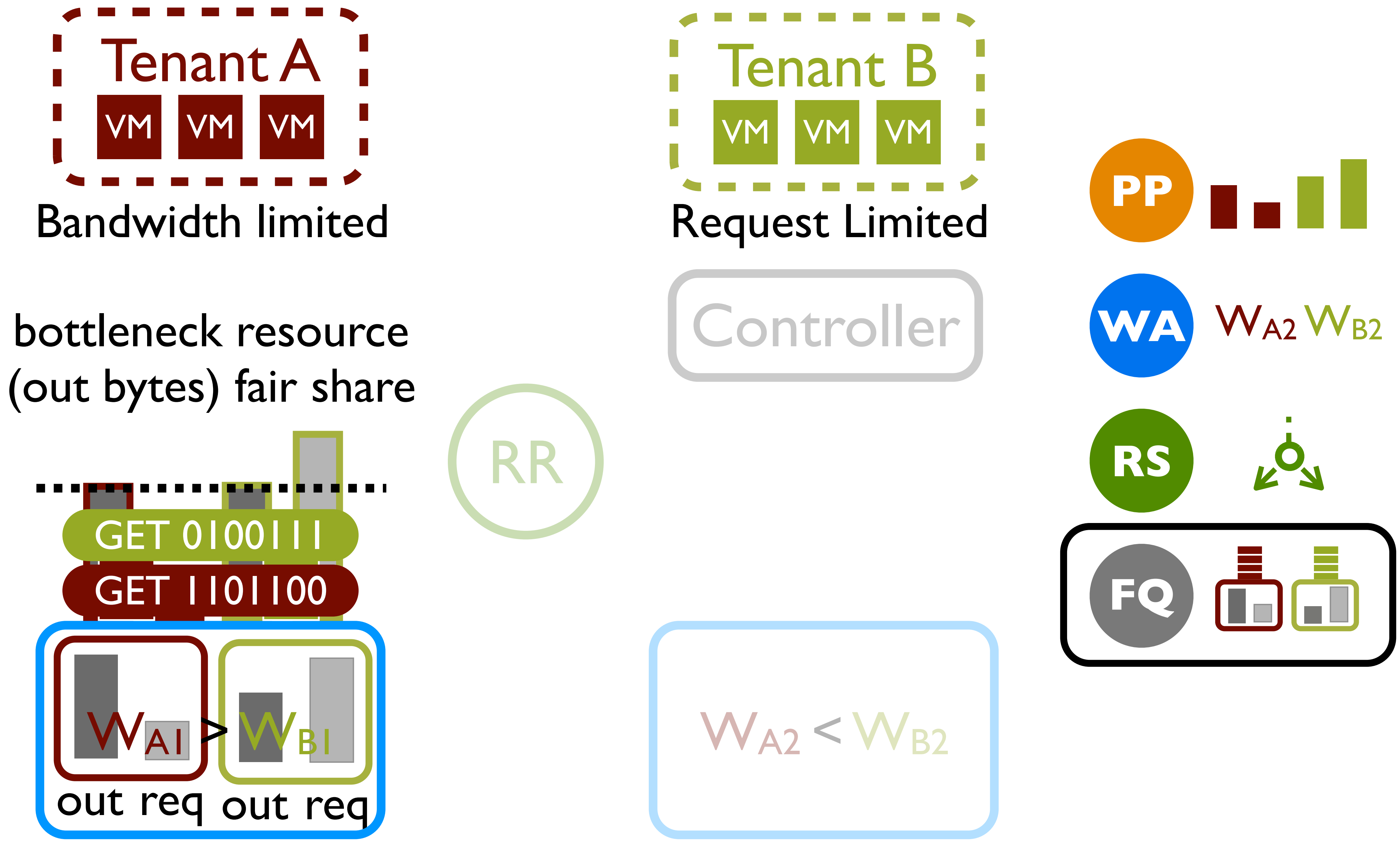


Pisces: Select Replicas By Local Weight

detect weight mismatch by request latency



Strawman: Queue Tenants By Single Resource



Tenant A

VM VM VM

Bandwidth limited

Tenant B

VM VM VM

Request Limited

Controller

bottleneck resource (out bytes) fair share

RR

GET 0100111

GET 1101100

$W_{A1} > W_{B1}$

out req out req

$W_{A2} < W_{B2}$

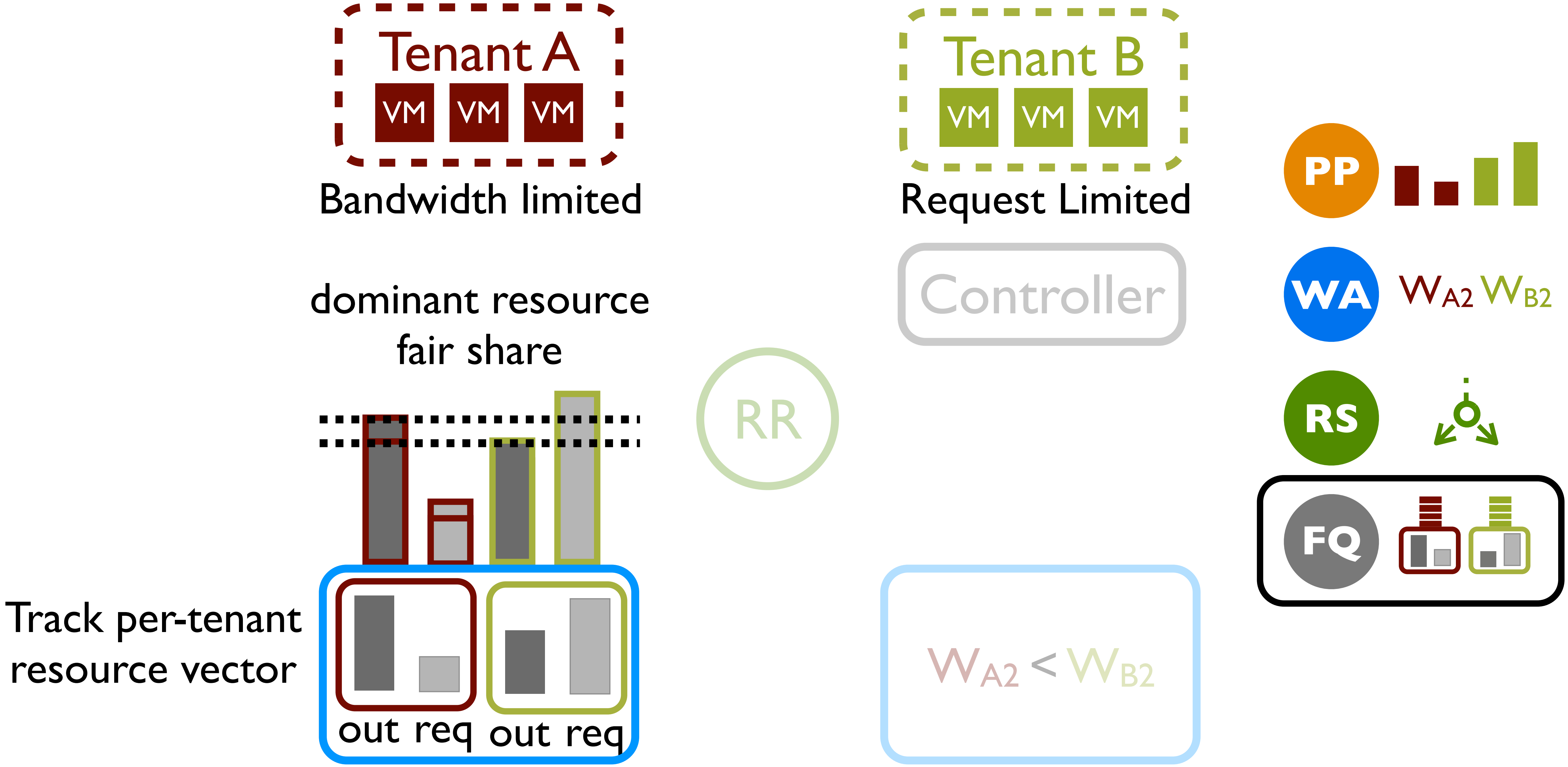
PP

WA W_{A2} W_{B2}

RS

FQ

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

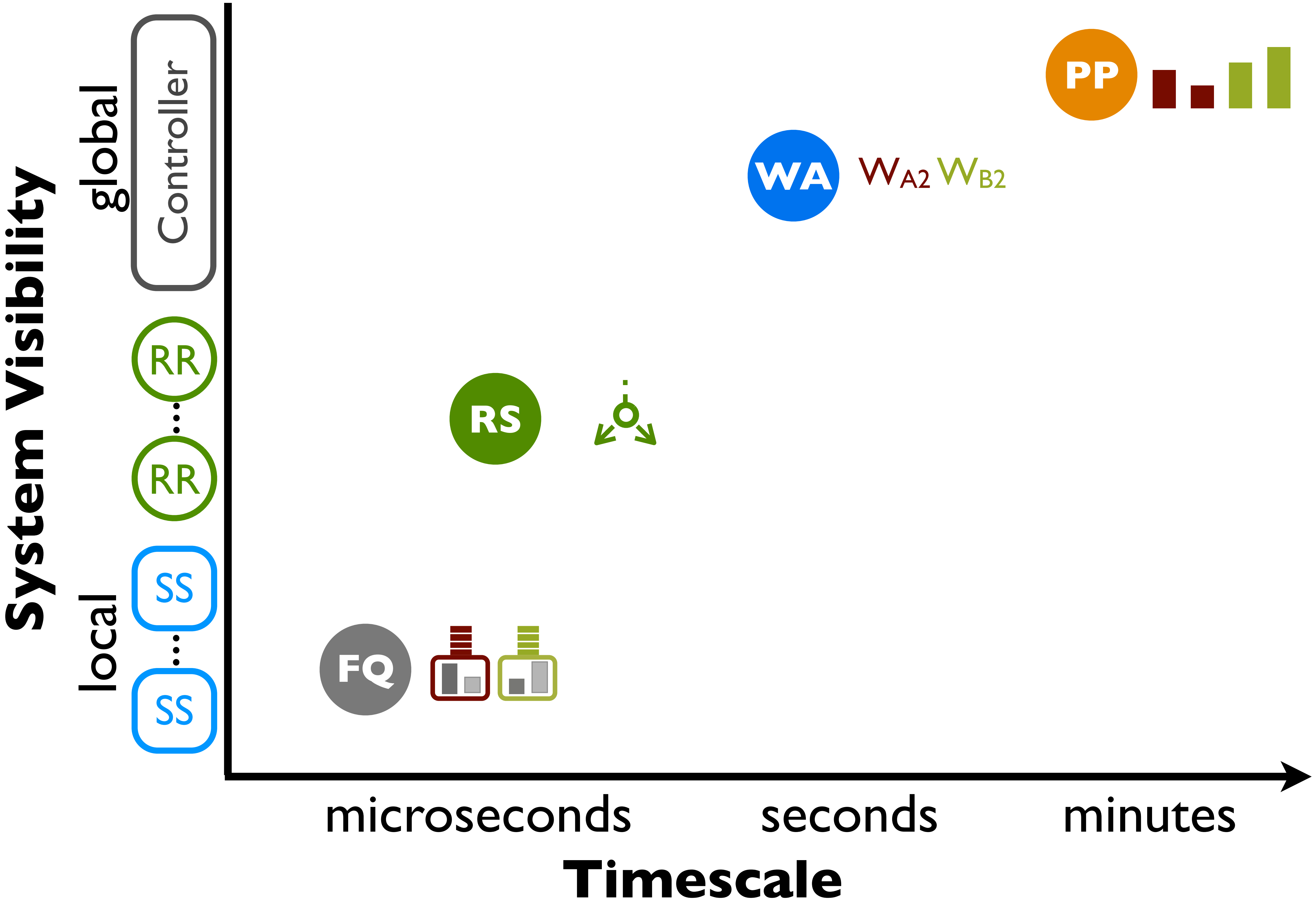
RR

Track per-tenant resource vector

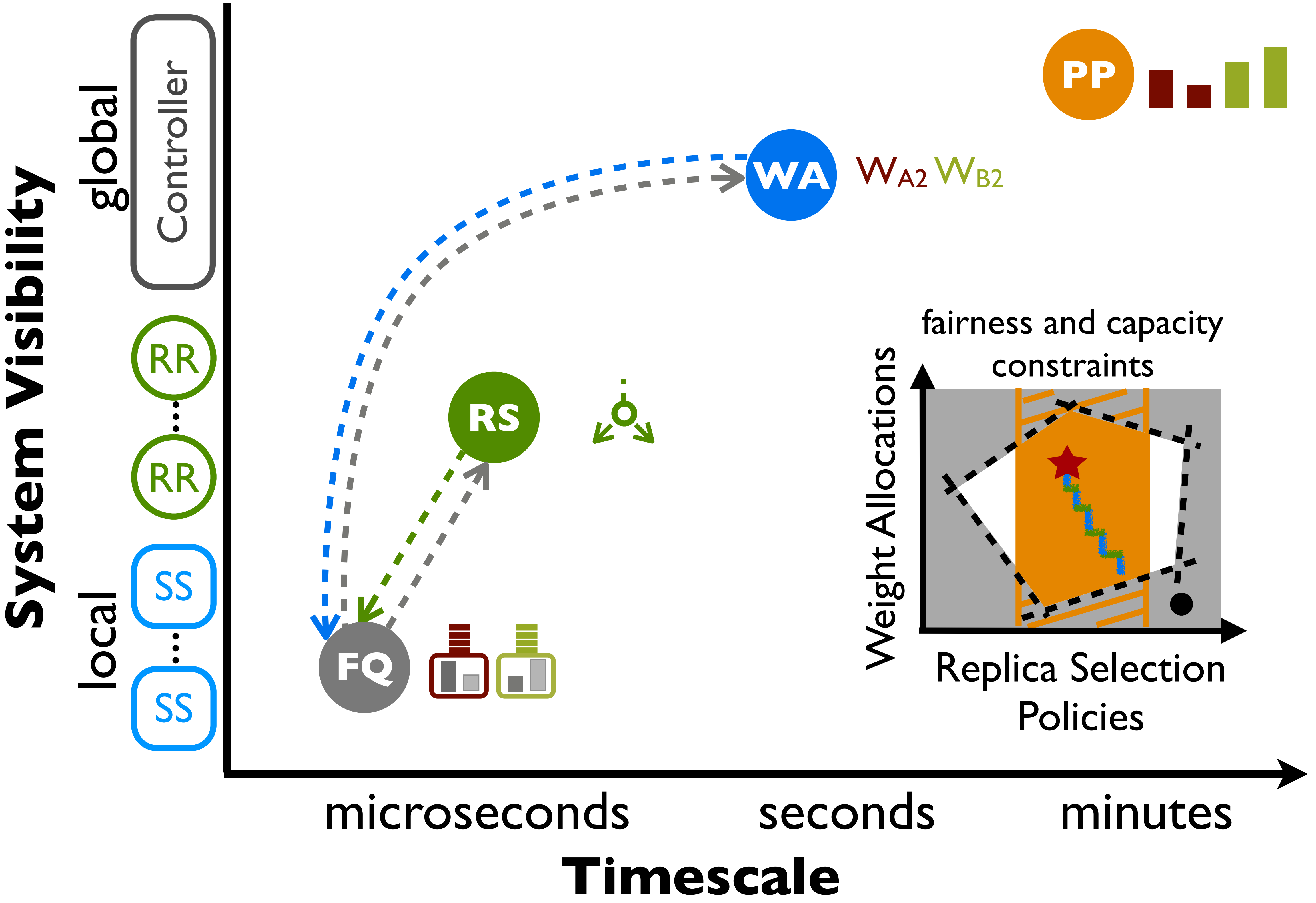
out req out req

$W_{A2} < W_{B2}$

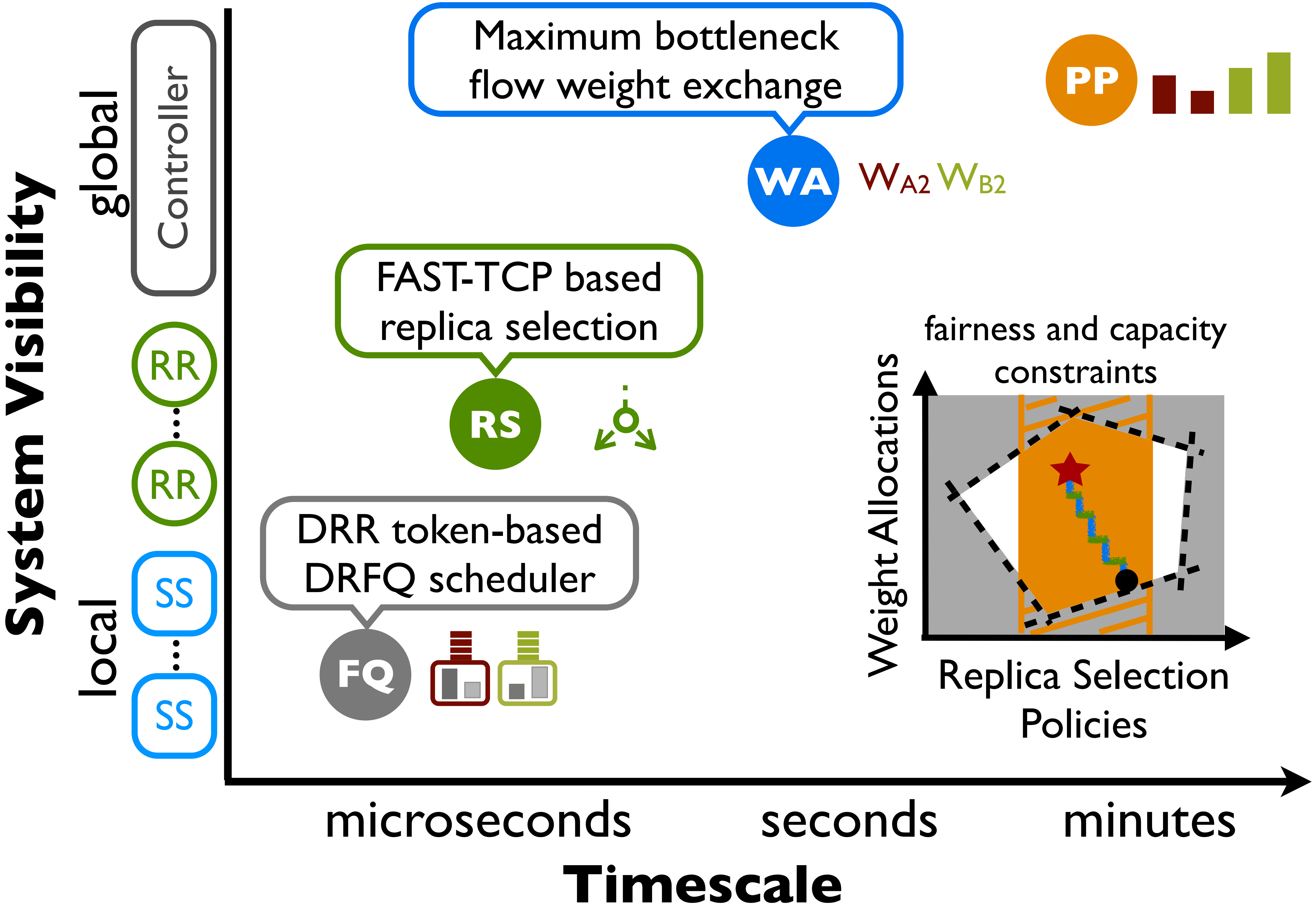
Pisces Mechanisms Solve For Global Fairness



Pisces Mechanisms Solve For Global Fairness



Pisces Mechanisms Solve For Global Fairness



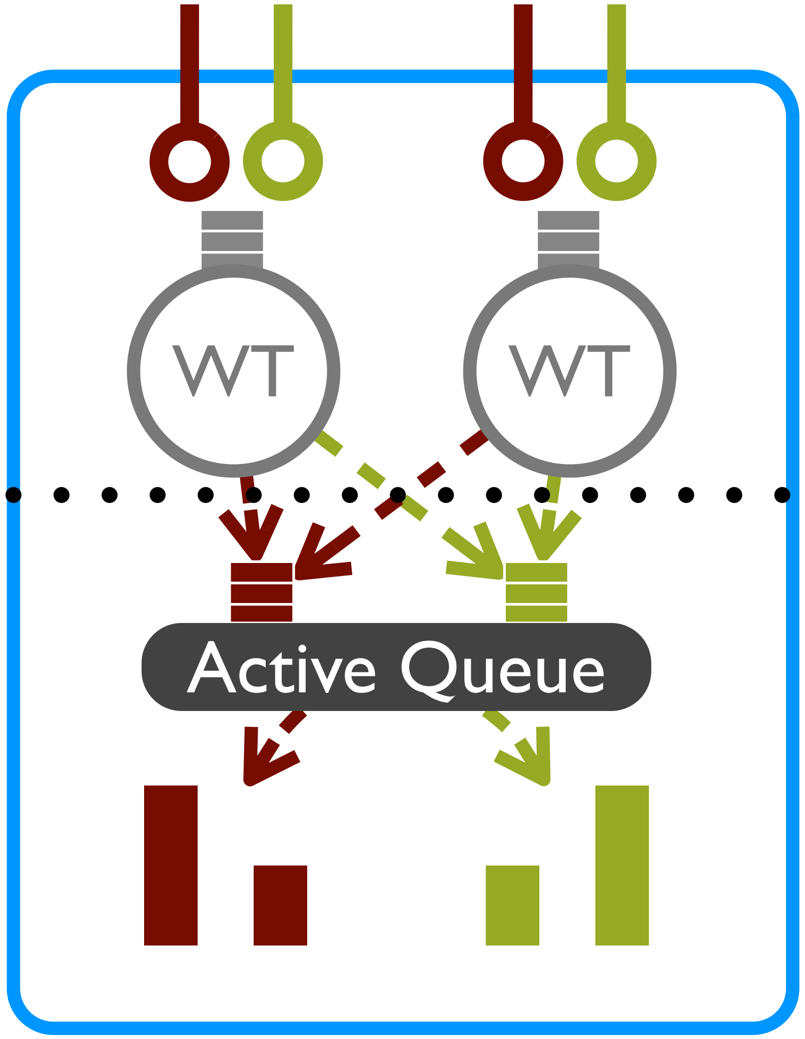
Maximum bottleneck flow weight exchange

FAST-TCP based replica selection

DRR token-based DRFQ scheduler

Implementing Fair Queuing

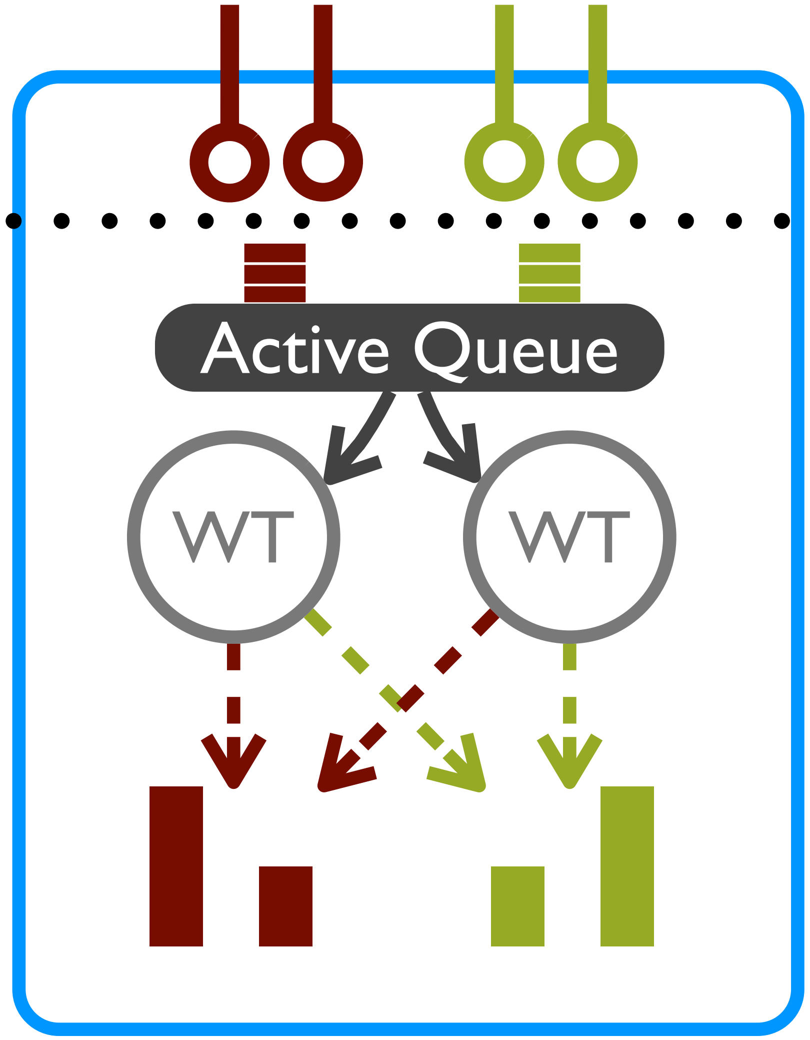
Per-request DWRR



Resources consumed before scheduling, violating fairness

Implementing Fair Queuing

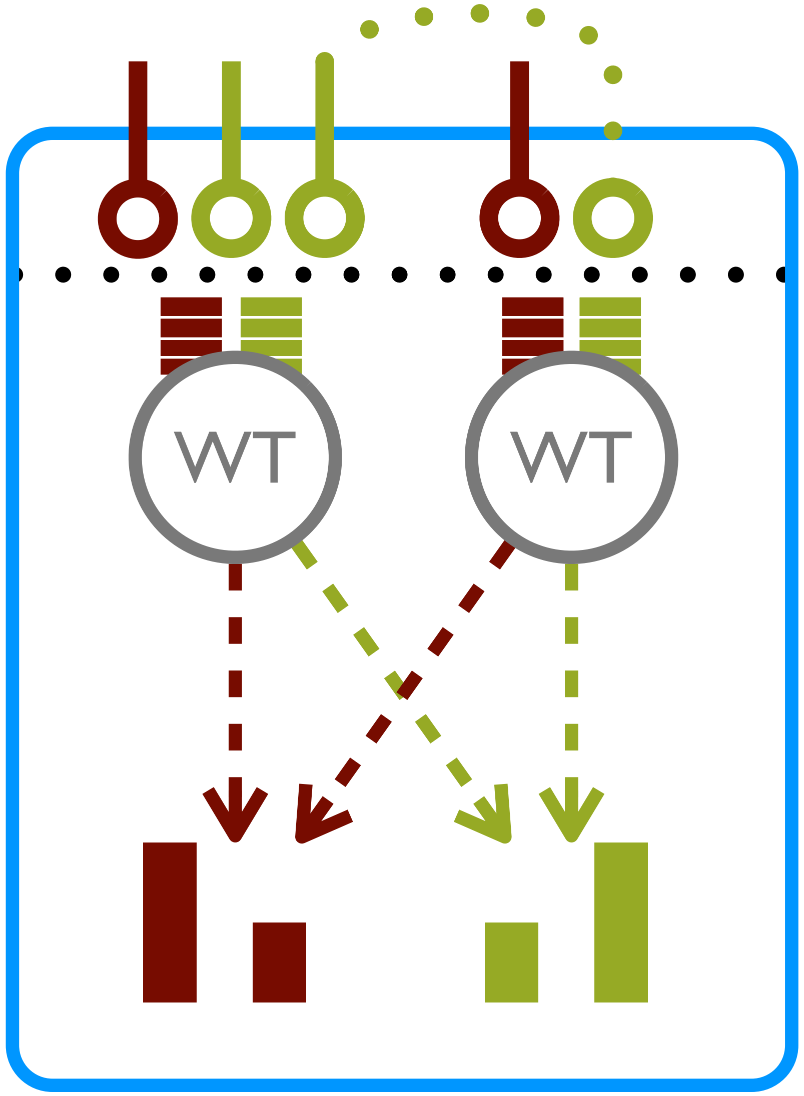
Per-connection DWRR



Lock contention leads to inefficiencies

Implementing Fair Queuing

Non-Blocking DWRR



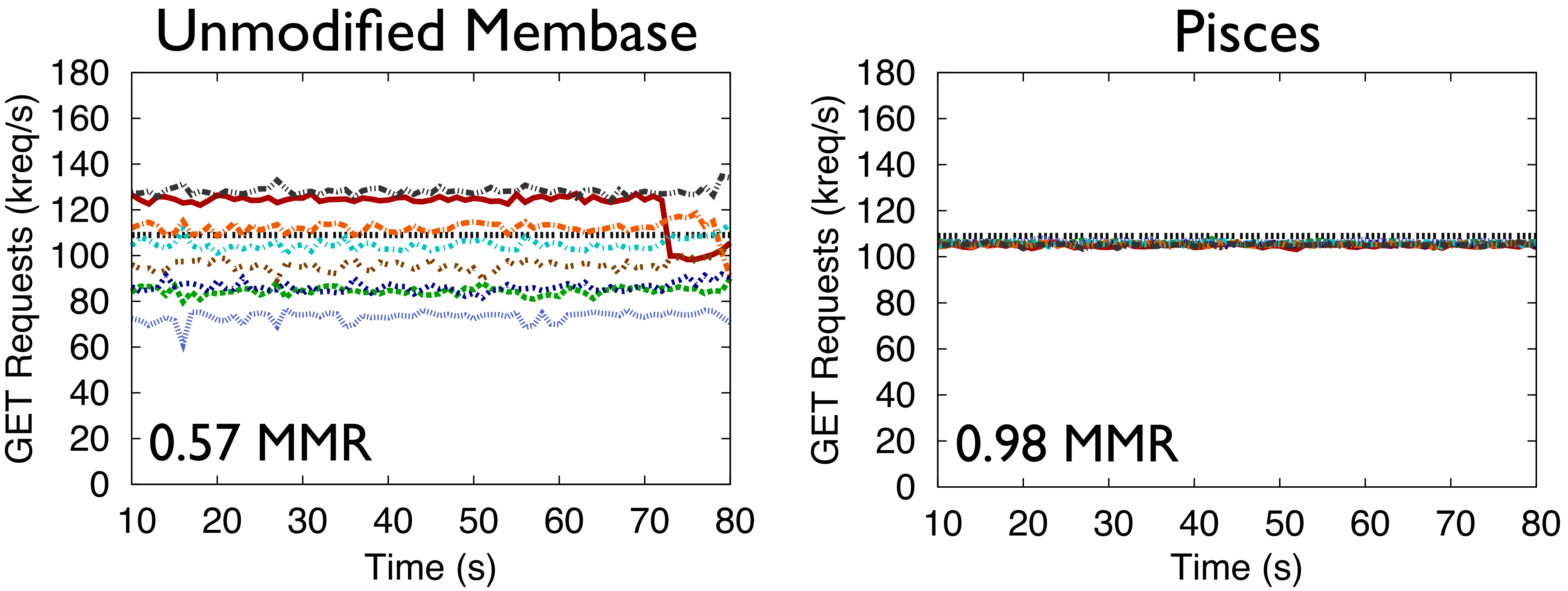
Distributed (Multi-core) scheduler optimizes throughput

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]

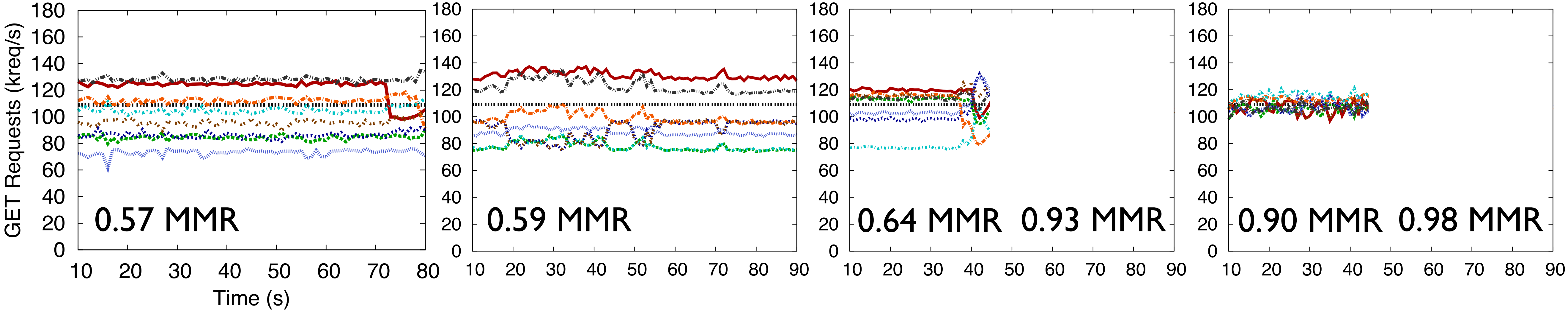
Each Pisces Mechanism Contributes to System-wide Fairness and Isolation

Unmodified Membase

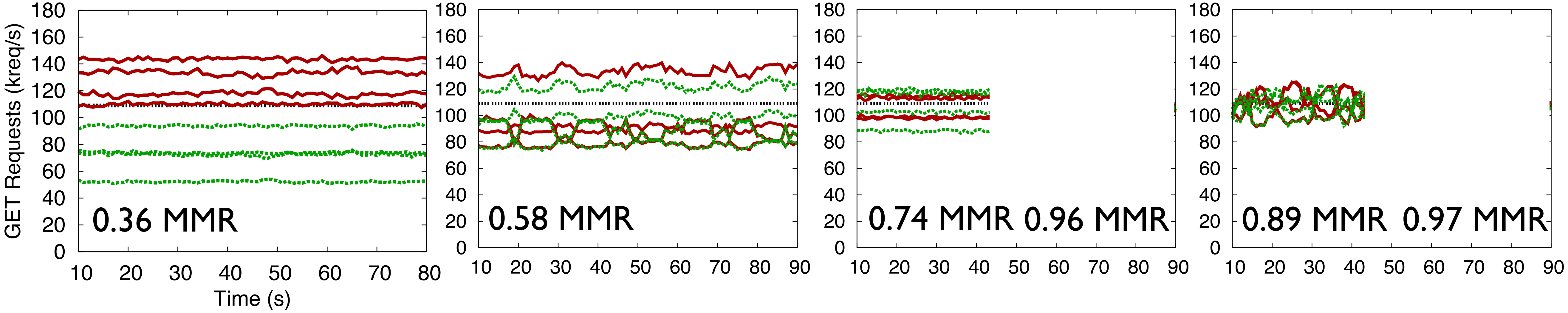
FQ

FQ PP WA

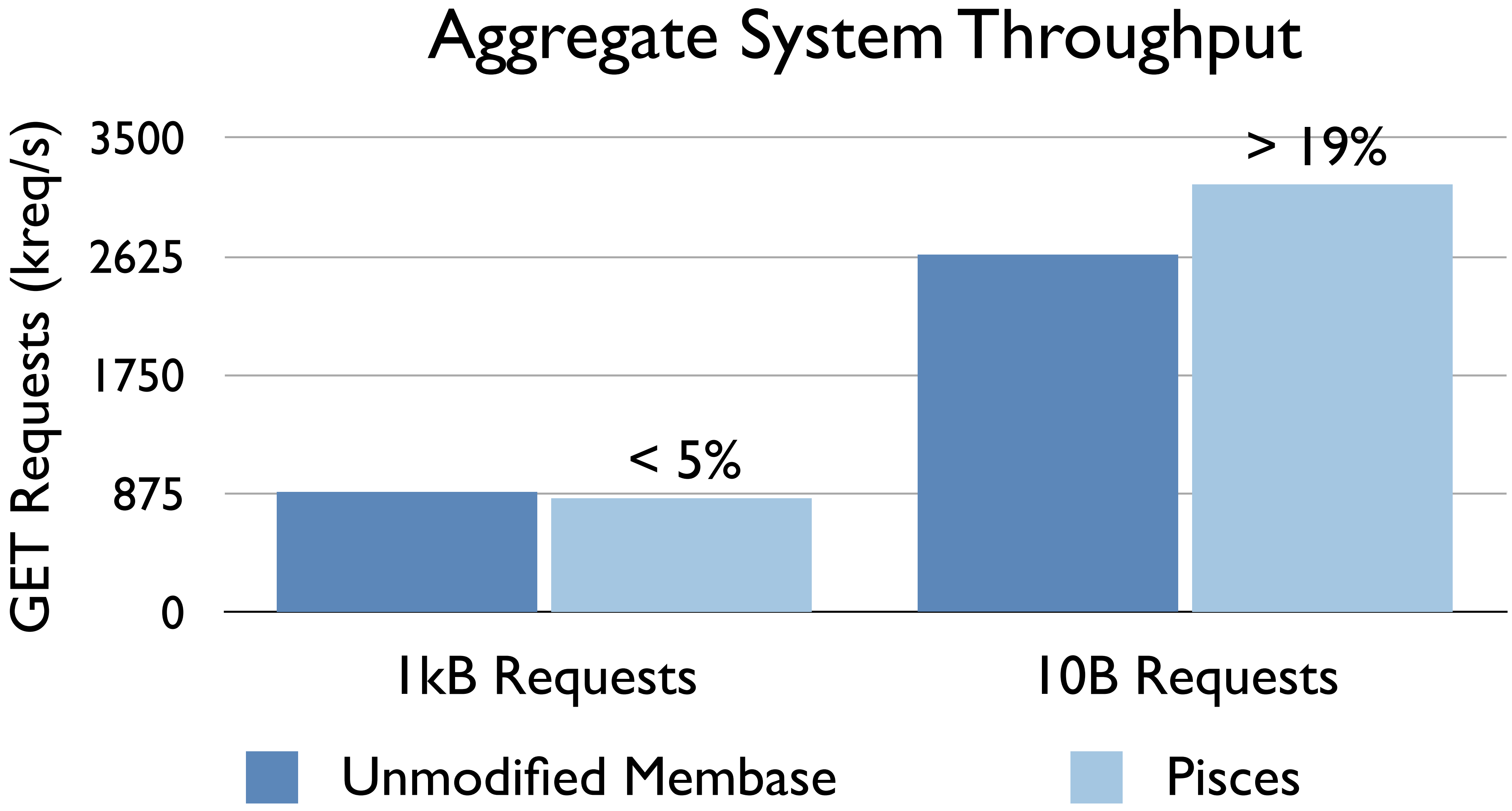
FQ PP RS WA



2x vs 1x demand



Pisces Imposes Low-overhead

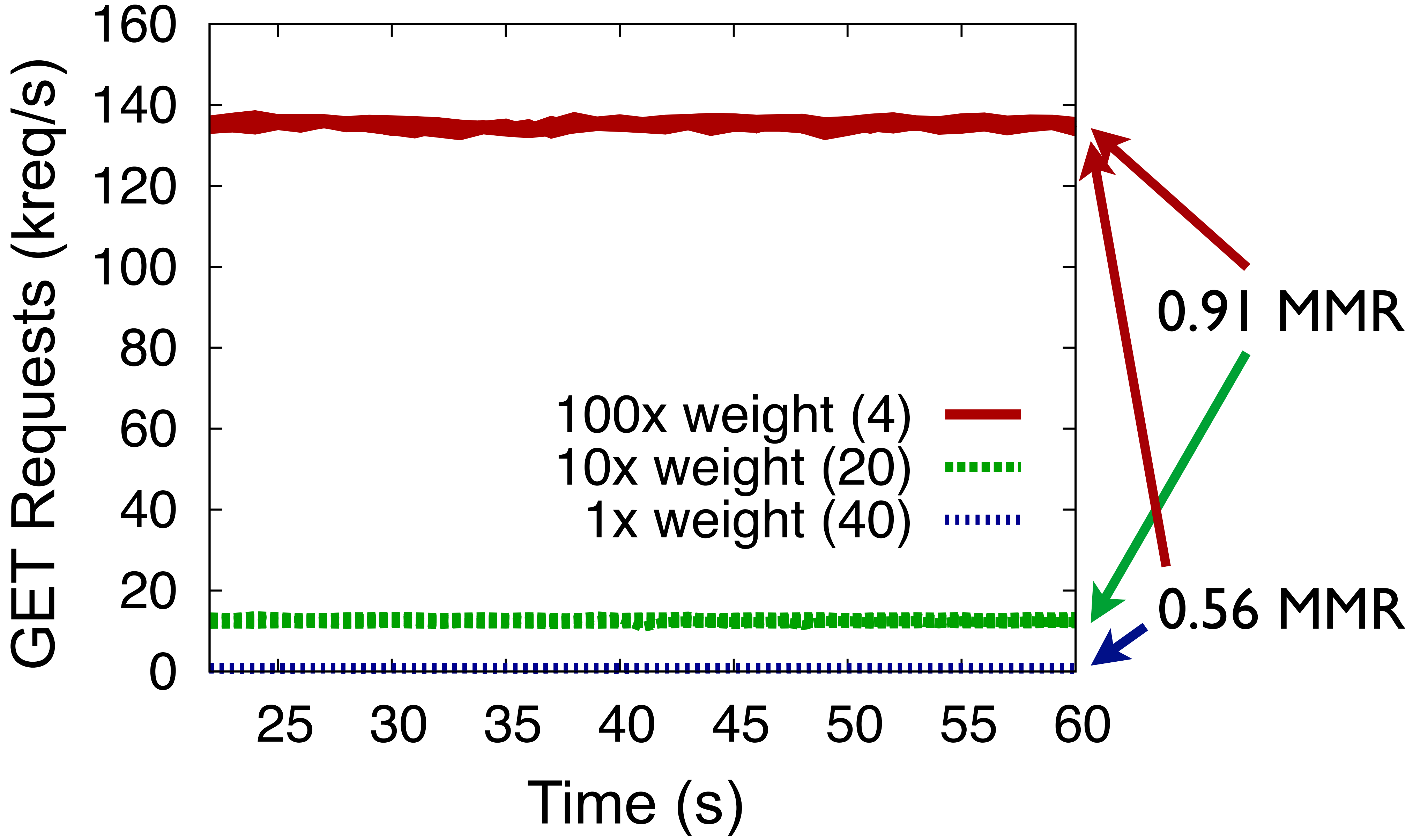


Pisces Achieves System-wide Weighted Fairness

0.98 MMR
4 heavy hitters

0.89 MMR
20 moderate demand

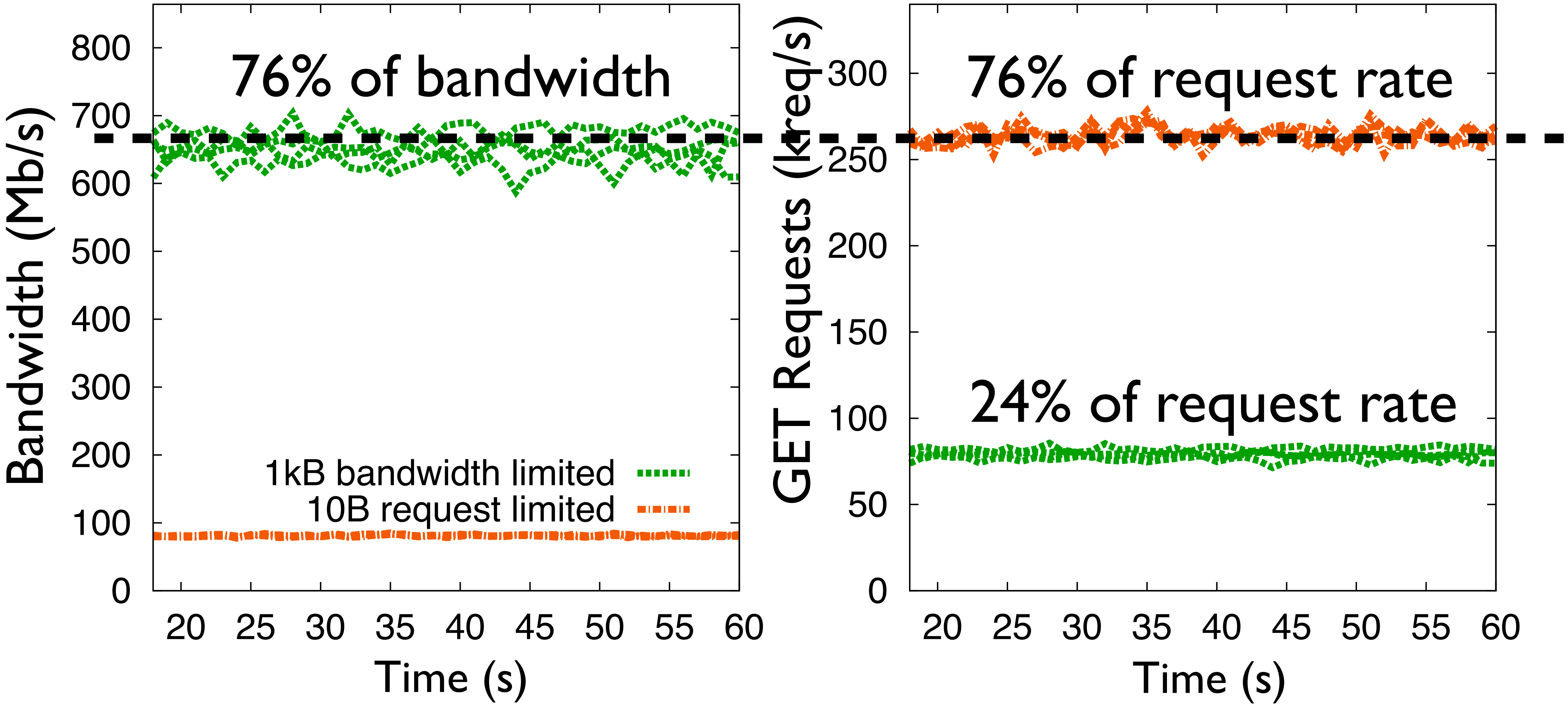
0.91 MMR
40 low demand



Pisces Achieves Dominant Resource Fairness

1kB workload
bandwidth limited

10B workload
request limited

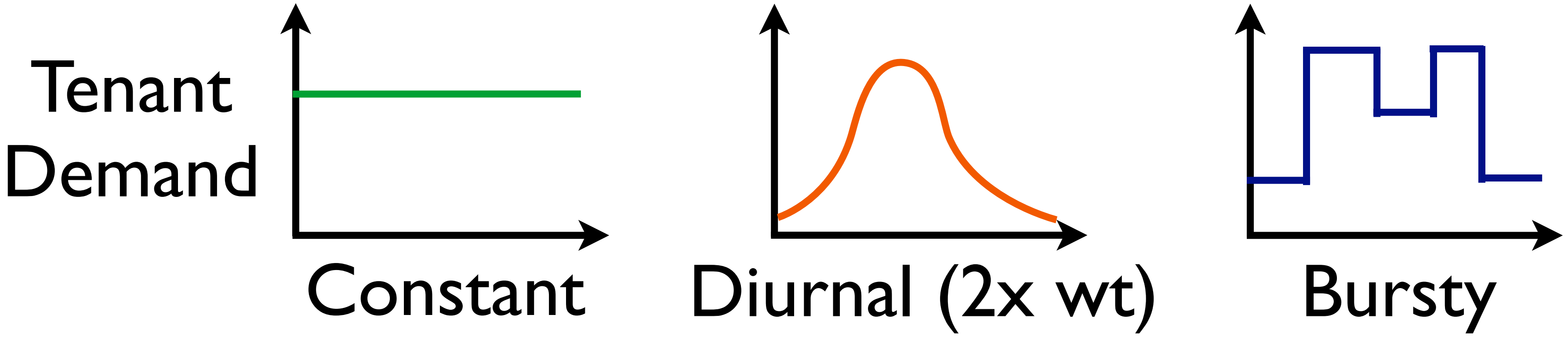


76% of bandwidth

76% of request rate

24% of request rate

Pisces Adapts to Dynamic Demand

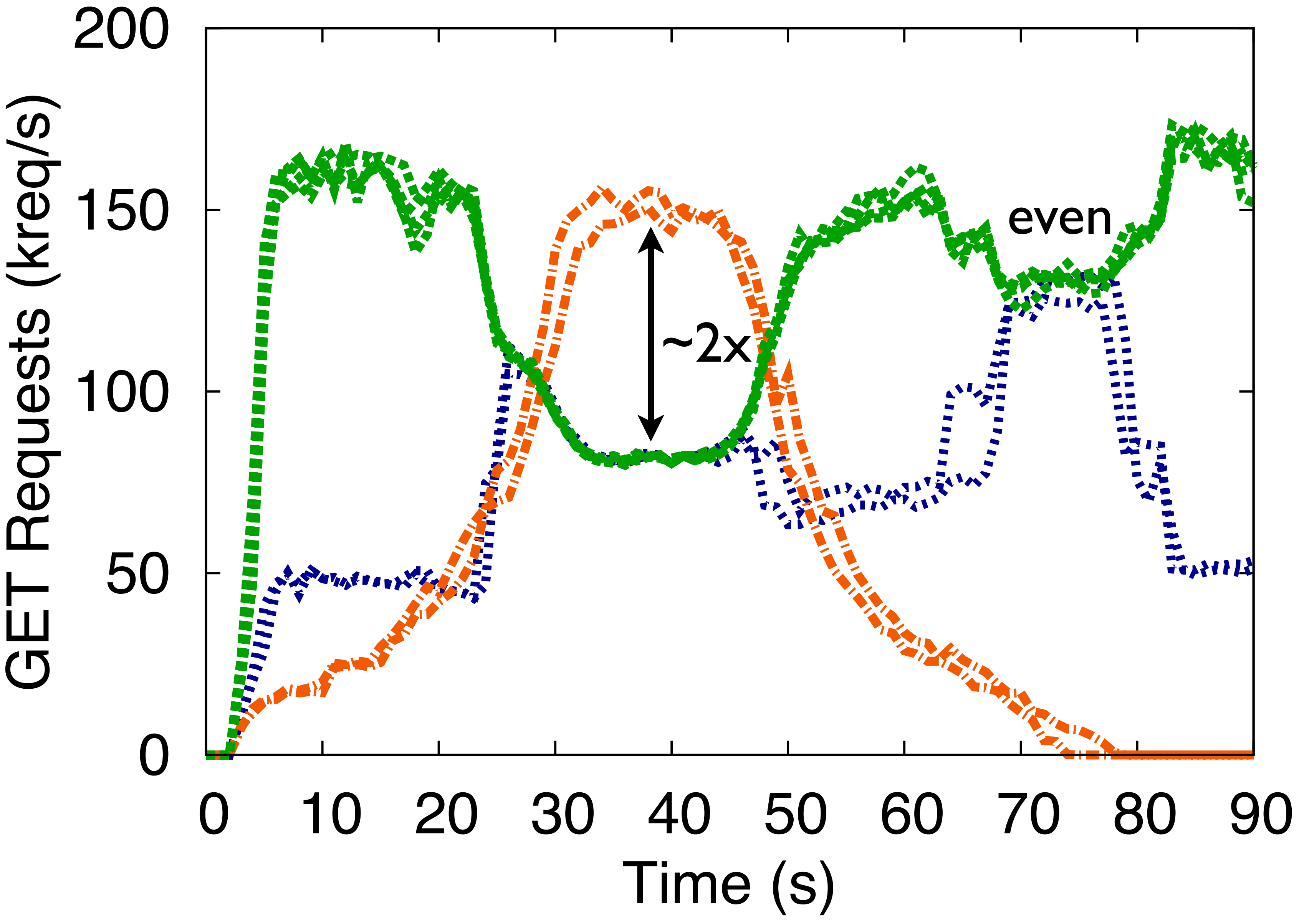


Tenant Demand

Constant

Diurnal (2x wt)

Bursty



~2x

even

Summary

- 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

- For more information:
  - OSDI ’12: System design, implementation, evaluation
  - Oper. Sys. Review ’13: Optimization decomposition
  - http://sns.cs.princeton.edu/projects/pisces/

Future Work: Generalized Fairness Framework

PP Partition Placement

- Include additional factors: migration cost, replicate vs. migrate, re-partitioning, MBF graph connectivity, resource workloads, etc.

WA Weight Allocation:

- Flexible allocation policies: SLO (Utility), Fairness, etc

RS Replication Selection

- Generic proxy-based service routing

FQ Fair-Queuing

- Library or in-kernel implementation
- Memory and Disk (IOPs) resources: Use ability to account for / handle partial requests rather than needing to predictive costs (e.g., for DB queries)
- Include reservations (min) and limits (max): DRF makes this harder

Future Work: Generalized Fairness Framework

PP Partition Placement

- Include additional factors: migration cost, replicate vs. migrate, re-partitioning, MBF graph connectivity, resource workloads, etc.

WA Weight Allocation:

- Flexible allocation policies: SLO (Utility), Fairness, etc

RS Replication Selection

- Generic proxy-based service routing

FQ Fair-Queuing

- Library or in-kernel implementation
- Memory and Disk (IOPs) resources: Use ability to account for / handle partial requests rather than needing to predictive costs (e.g., for DB queries)
- Include reservations (min) and limits (max): DRF makes this harder

Related Work

	Resource	Scope	MT	Fairness	Policy	Resolution	Mechanism
Parda	IOPs	SAN access	Yes*	Per-Node	Proportional	Request	FAST-TCP
mClock	IOPs	Block Storage	Yes	Per-VM	FQ/Res/Lim	Request	VT-scheduler
Maestro	IOPs	Disk Array	Yes	Per-App	SLO	Request	linear model, LP allocator
FAST	Disk	Block Storage	No	Per-Workload	Insulation	Workload	storage layout, routing
Argon	Mem/IOPs	Single Node	Yes	Per-Client	Insulation	Request	mem man, disk
Cake	CPU/Disk	HBase/HDFS	Yes (2)	Per-Client	SLO	Request	FQ, additive allocation
Mesos	CPU/Mem	IaaS	Yes	Per-Tenant	DRF	Task	DRF allocator
Auto control	CPU/IOPs	IaaS	Yes	Per-App	SLO	VM	linear model, LP allocator
DRFQ	Net/CPU/Mem	Single Switch	Yes*	Per-Flow	DRF	Packet	VT-scheduler
Pisces	Net/~CPU	K-V Storage	Yes	Per-Tenant	Max/Min (+ LocalDRF)	Request	PP + WA + RS + FQ