Skupper sizing guide

Overview

This guide examines Skupper performance at varying allocations of router CPU. It uses constant throughput benchmarks to gauge latencies for different kinds of workloads.

The general setup:

- The benchmark tools are containerized client-server applications.
- There are two Skupper sites linked to form a network. The benchmark client runs in one site, the server in the other.
- Each site is running in its own Kubernetes cluster. The clusters are in the same data center.
- We run the benchmarks at constant throughput levels and capture latency metrics as we scale router CPU.

Date of testing: September 2023

Test environment

These tests were conducted using two clusters in the same data center in order to minimize latency from layer 3 networking and geographical distance.

Cloud provider: IBM
Zone: Washington DC 1 (us-east-1)
Kubernetes flavor: OpenShift
OpenShift version: 4.12.30_1556
Worker nodes: 4 (per cluster)

Node flavor: bx2.8x32 (Public Cloud Reference page)
Node CPU cores: 4
Node vCPUs: 8
Node RAM: 32 GiB
Node bandwidth cap: 16 Gbps
Node OS: RHEL 8.8

Skupper configuration

Skupper version: 1.4.2

Protocol: TCP

HTTP/2 (h2load)

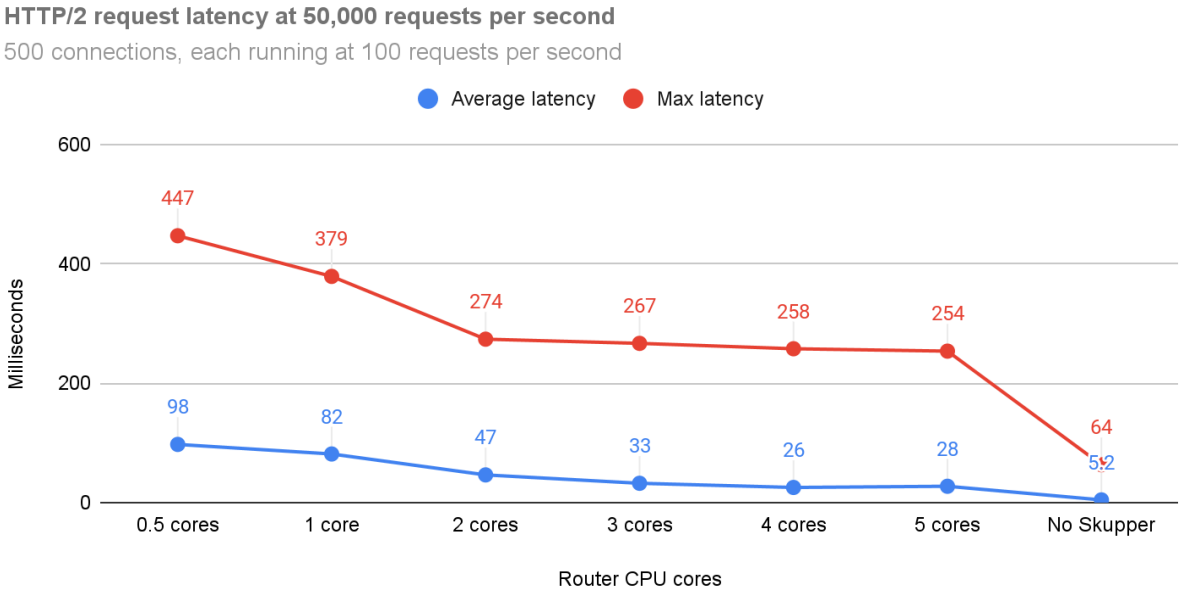
Benchmark tool: [h2load]
Benchmark scripts: https://github.com/skupperproject/benchdog/tree/main/h2load

H2load doesn't expose latencies by percentile, so we are using average and max here. Note that the max latency value is highly variable run to run.

50,000 requests per second (500 connections)

Latency figures are in milliseconds. The "No Skupper" case uses a Kubernetes Service of type LoadBalancer. Unlike Skupper, it does not TLS encrypt inter-cluster traffic.

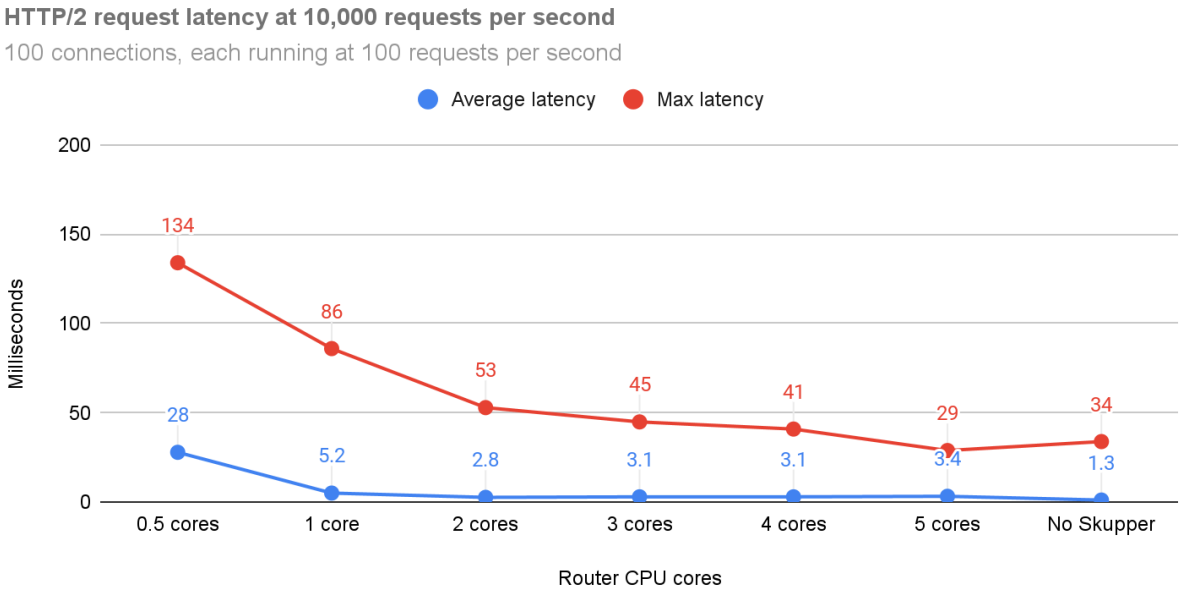
	0.5 cores	1 core	2 cores	3 cores	4 cores	5 cores	No Skupper
Average latency	98	82	47	33	26	28	5.2
Max latency	447	379	274	267	258	254	64



10,000 requests per second (100 connections)

Latency figures are in milliseconds. The “No Skupper” case uses a Kubernetes Service of type LoadBalancer. Unlike Skupper, it does not TLS encrypt inter-cluster traffic.

	0.5 cores	1 core	2 cores	3 cores	4 cores	5 cores	No Skupper
Average latency	28	5.2	2.8	3.1	3.1	3.4	1.3
Max latency	134	86	53	45	41	29	34

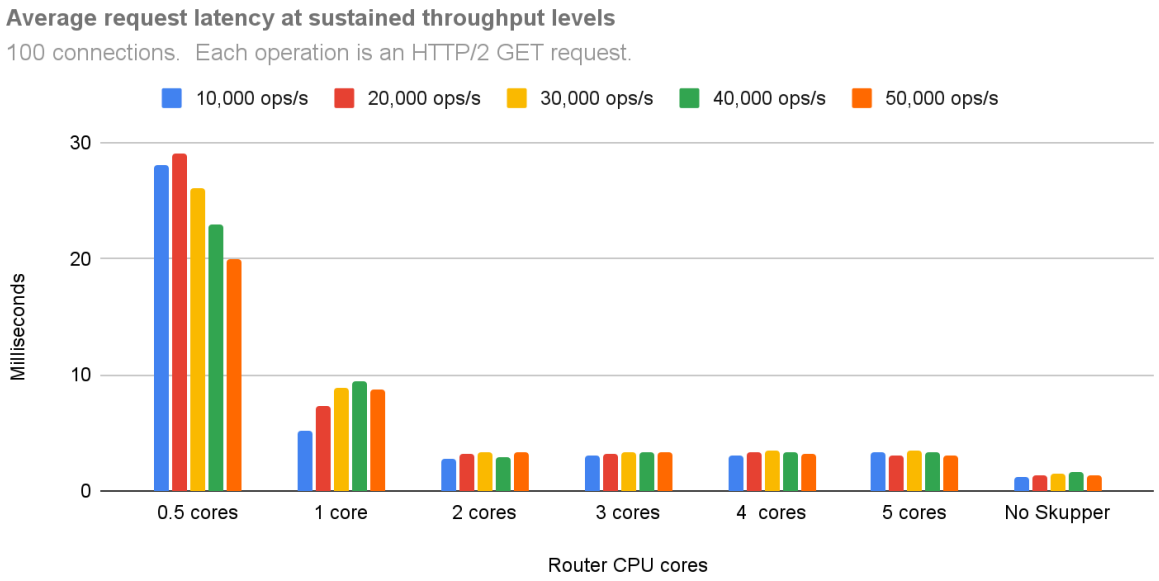


Average request latency at sustained throughput levels

Latency figures are in milliseconds. The “No Skupper” case uses a Kubernetes Service of type LoadBalancer. Unlike Skupper, it does not TLS encrypt inter-cluster traffic.

These tests were all conducted with 100 concurrent connections.

	0.5 cores	1 core	2 cores	3 cores	4 cores	5 cores	No Skupper
10,000 ops/s	28	5.2	2.8	3.1	3.1	3.4	1.3
20,000 ops/s	29	7.4	3.2	3.2	3.4	3.1	1.4
30,000 ops/s	26	8.9	3.4	3.3	3.5	3.5	1.5
40,000 ops/s	23	9.4	3.0	3.3	3.3	3.3	1.6
50,000 ops/s	20	8.8	3.3	3.3	3.2	3.1	1.4



PostgreSQL (pgbench)

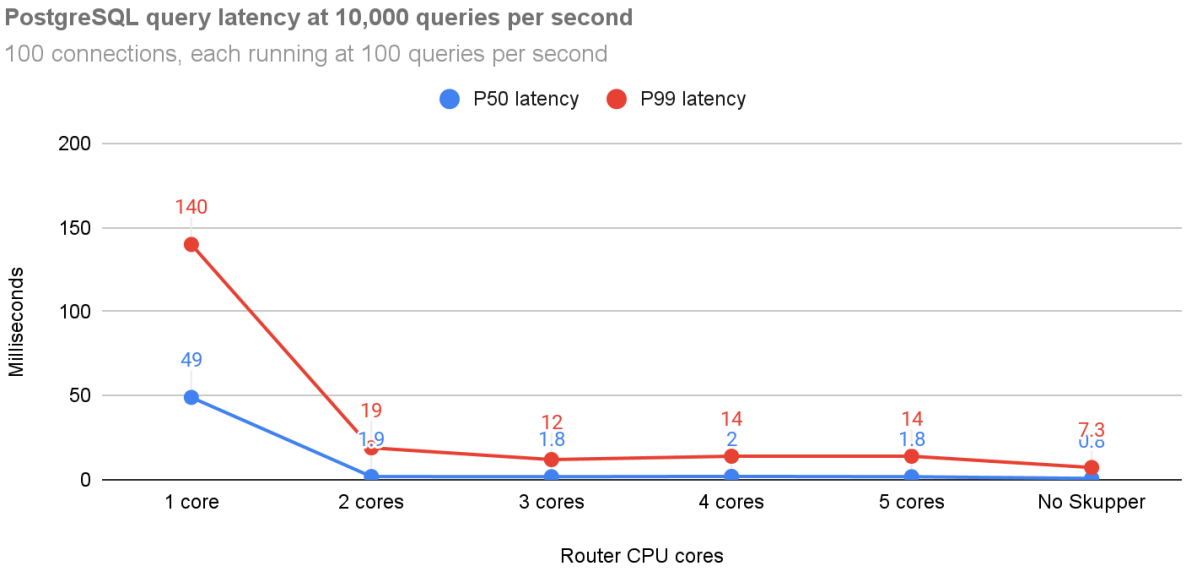
Benchmark tool: pgbench
Benchmark scripts: https://github.com/skupperproject/benchdog/tree/main/pgbench

Pgbench is notably sensitive to network latency. In order to sustain higher levels of throughput, Skupper must have sufficient CPU to keep latencies low.

10,000 queries per second (100 connections)

Latency figures are in milliseconds. Note that the latencies include operation start delays ("schedule lag" in pgbench). The "No Skupper" case uses a Kubernetes Service of type LoadBalancer. Unlike Skupper, it does not TLS encrypt inter-cluster traffic.

	1 core	2 cores	3 cores	4 cores	5 cores	No Skupper
P50 latency	49	1.9	1.8	2.0	1.8	0.8
P99 latency	140	19	12	14	14	7.3

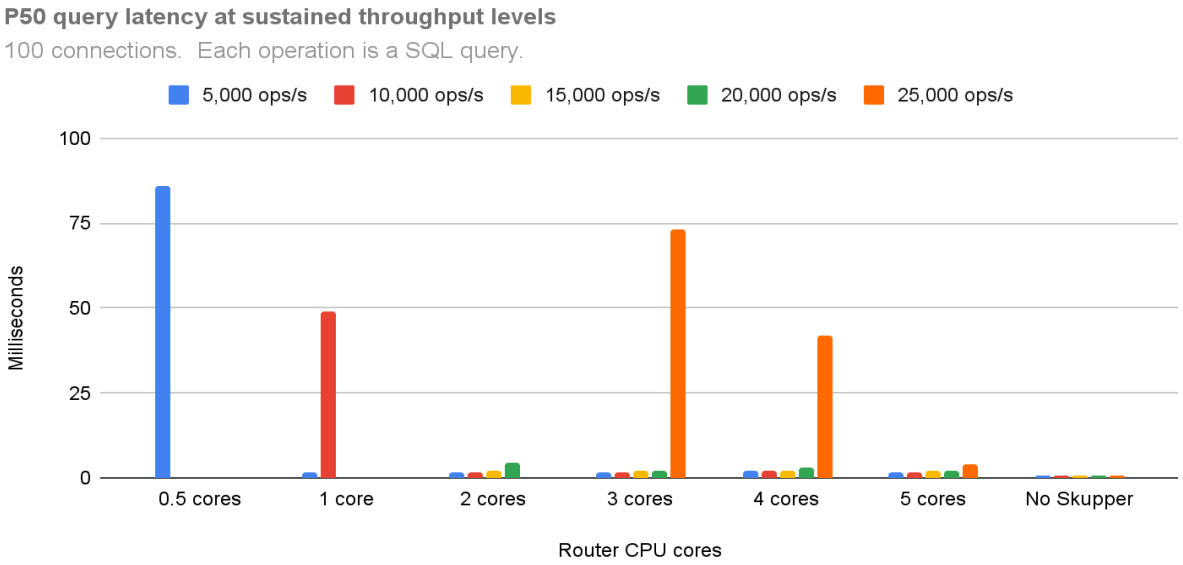


P50 query latency at sustained throughput levels

Latency figures are in milliseconds. Note that the latencies include operation start delays ("schedule lag" in pgbench). The "No Skupper" case uses a Kubernetes Service of type LoadBalancer. Unlike Skupper, it does not TLS encrypt inter-cluster traffic.

These tests were all conducted with 100 concurrent connections. Cells without values indicate cases where the throughput level could not be sustained.

	0.5 cores	1 core	2 cores	3 cores	4 cores	5 cores	No Skupper
5,000 ops/s	86	1.9	1.8	1.7	2.1	1.8	0.9
10,000 ops/s		49	1.9	1.8	2.0	1.8	0.8
15,000 ops/s			2.1	2.0	2.4	2.0	0.8
20,000 ops/s			4.7	2.4	3.1	2.3	0.9
25,000 ops/s				73	42	4.1	0.9



Messaging (qbench)

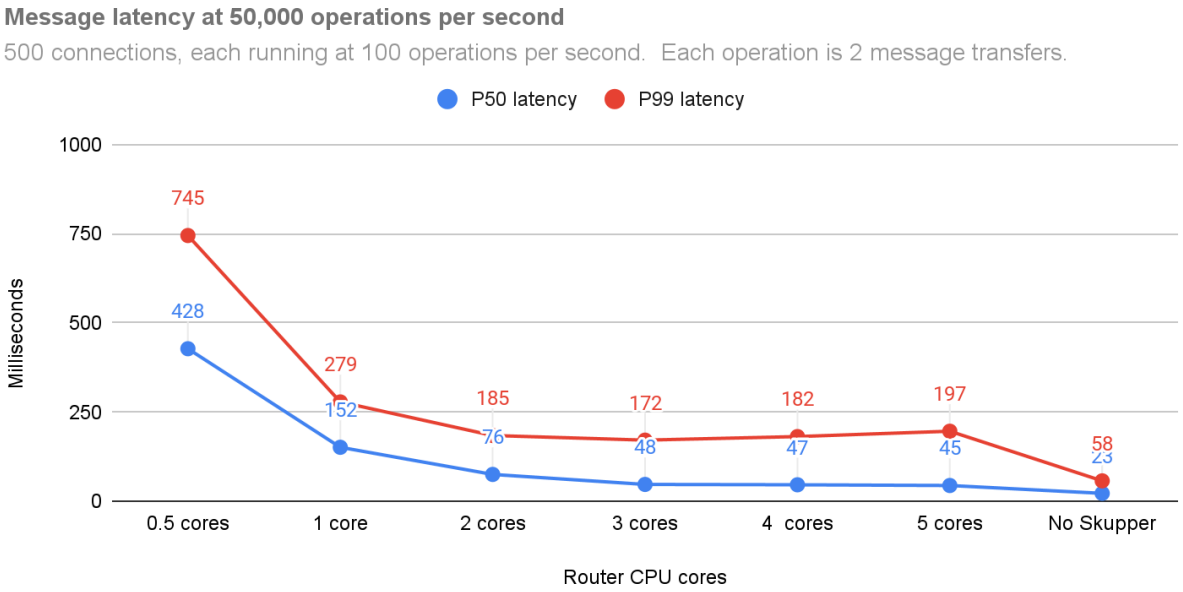
Benchmark tool: [qbench]
Benchmark scripts: https://github.com/skupperproject/benchdog/tree/main/qbench

In contrast to pgbench, which is latency sensitive, qbench is latency tolerant. All operations are fully asynchronous.

50,000 operations per second (500 connections)

Latency figures are in milliseconds. The “No Skupper” case uses a Kubernetes Service of type LoadBalancer. Unlike Skupper, it does not TLS encrypt inter-cluster traffic.

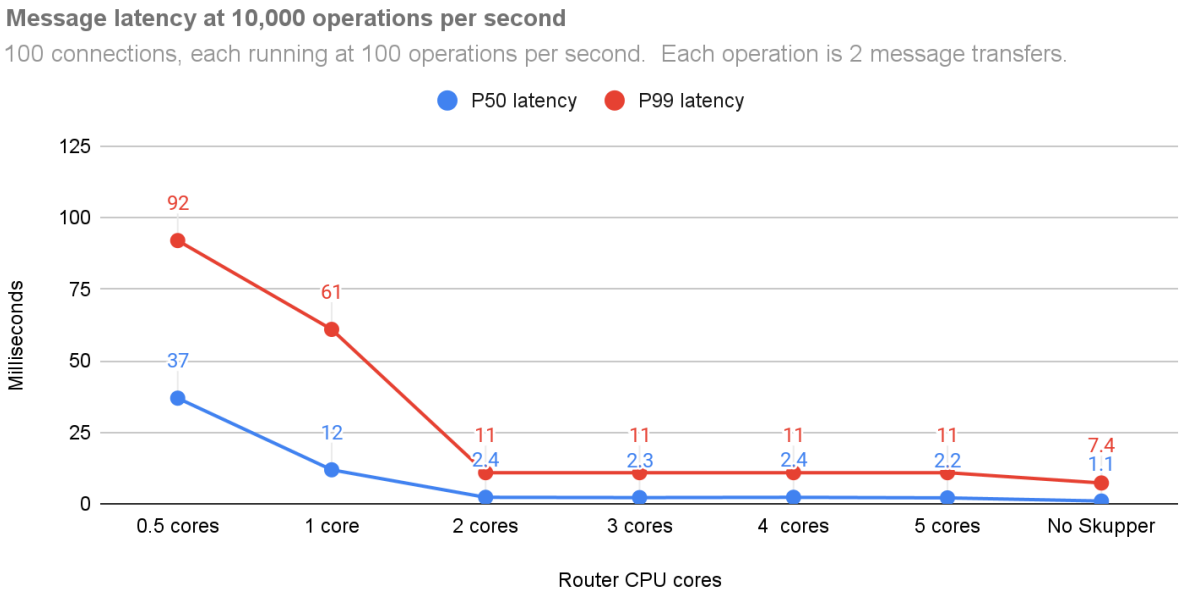
	0.5 cores	1 core	2 cores	3 cores	4 cores	5 cores	No Skupper
P50 latency	428	152	76	48	47	45	23
P99 latency	745	279	185	172	182	197	58



10,000 operations per second (100 connections)

Latency figures are in milliseconds. The “No Skupper” case uses a Kubernetes Service of type LoadBalancer. Unlike Skupper, it does not TLS encrypt inter-cluster traffic.

	0.5 cores	1 core	2 cores	3 cores	4 cores	5 cores	No Skupper
P50 latency	37	12	2.4	2.3	2.4	2.2	1.1
P99 latency	92	61	11	11	11	11	7.4

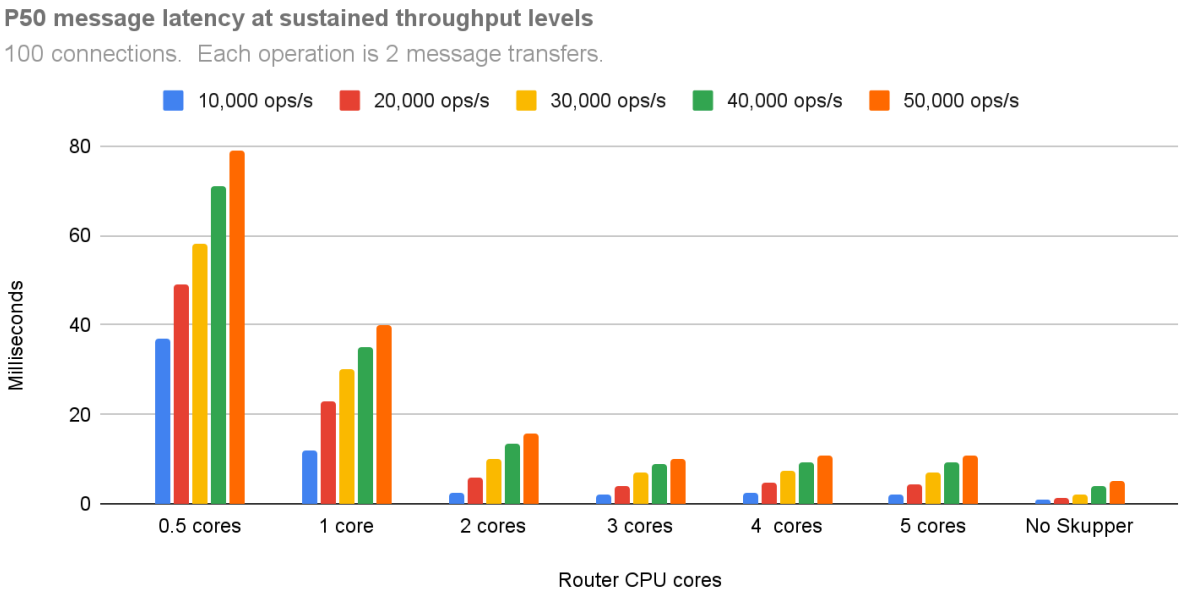


P50 message latency at sustained throughput rates

Latency figures are in milliseconds. The “No Skupper” case uses a Kubernetes Service of type LoadBalancer. Unlike Skupper, it does not TLS encrypt inter-cluster traffic.

These tests were all conducted with 100 concurrent connections.

	0.5 cores	1 core	2 cores	3 cores	4 cores	5 cores	No Skupper
10,000 ops/s	37	12	2.4	2.3	2.4	2.2	1.1
20,000 ops/s	49	23	5.9	4.1	4.7	4.3	1.4
30,000 ops/s	58	30	10.2	7.0	7.3	7.2	2.2
40,000 ops/s	71	35	13.5	9.0	9.5	9.5	4.0
50,000 ops/s	79	40	15.9	10	11	11	5.1



Kafka Benchmark

1. The Benchmark

- A containerized Kafka benchmark tool using the Kafka Java client and the Strimzi operator was run on two IBM clusters, connected by Skupper.
- The benchmark was run with CPU settings varying from 0.5 to 4.0, in steps of 0.5.

- CPU setting applies to three of the Virtual Application Network's components:

    - the Skupper router on the client cluster
    - the Skupper router on the server cluster
    - the Skupper edge-router 'proxy' on the server cluster

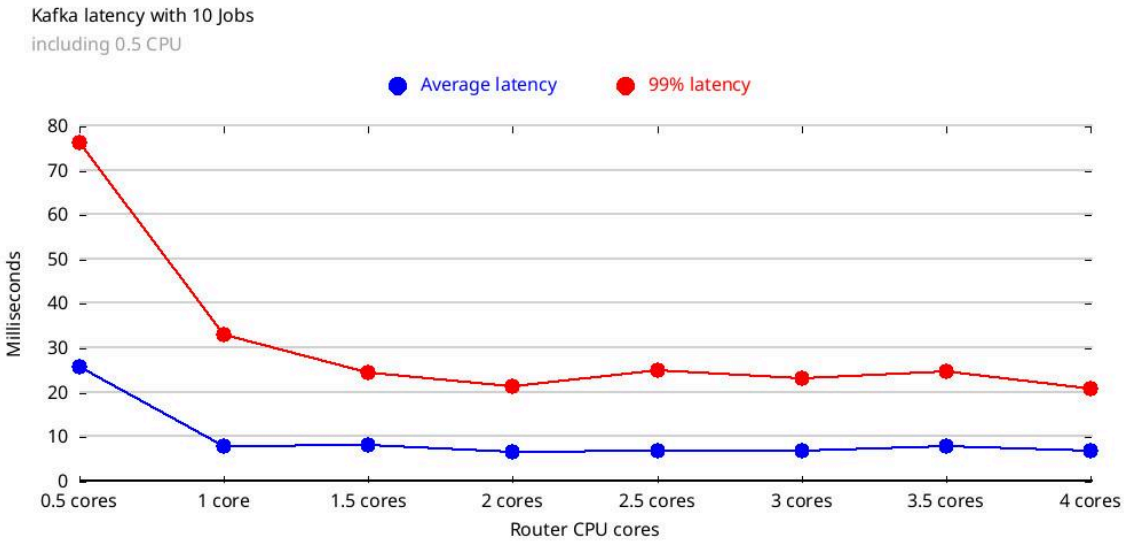
- The benchmark is run with 10 and 100 concurrent jobs
- Each job runs 100 operations per second
- Operation latency is measured and graphed

2. Kafka Latency Outliers

Kafka occasionally produces extreme latency outliers that make it difficult to determine meaningful Skupper sizing guidelines. To work around this problem, I ran each benchmark multiple times and selected runs that were not affected by the occasional extreme outliers. The resulting graphs provide a more consistent picture of appropriate Skupper sizing.

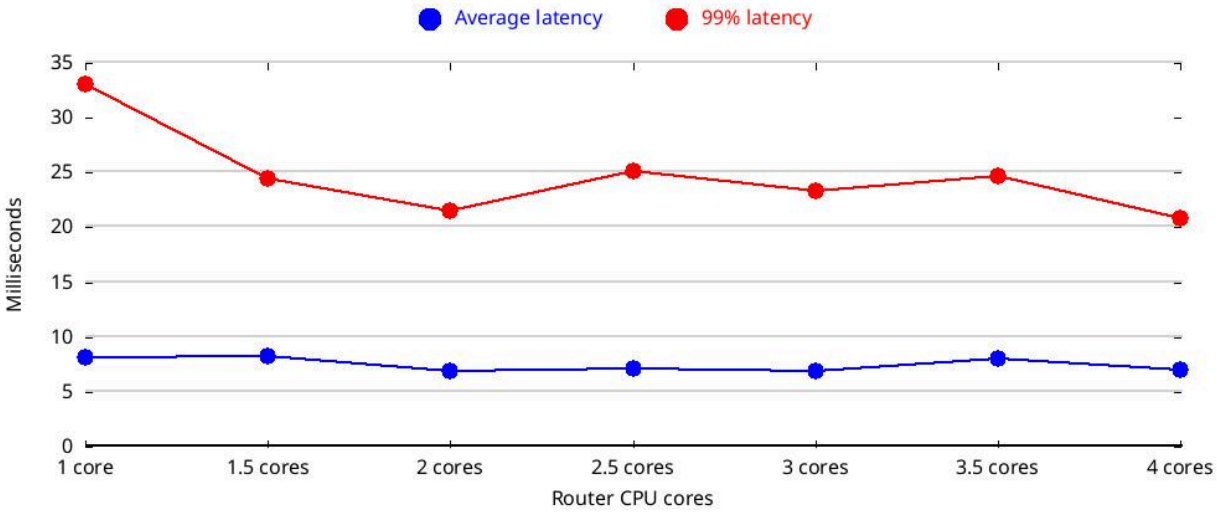
3. Benchmark Results

Two graphs are presented for each set of tests: one with the 0.5 CPU data point present, and one without. The latencies for the 0.5 CPU tests are so high that it is difficult to see meaningful differences in the higher-CPU measurements.



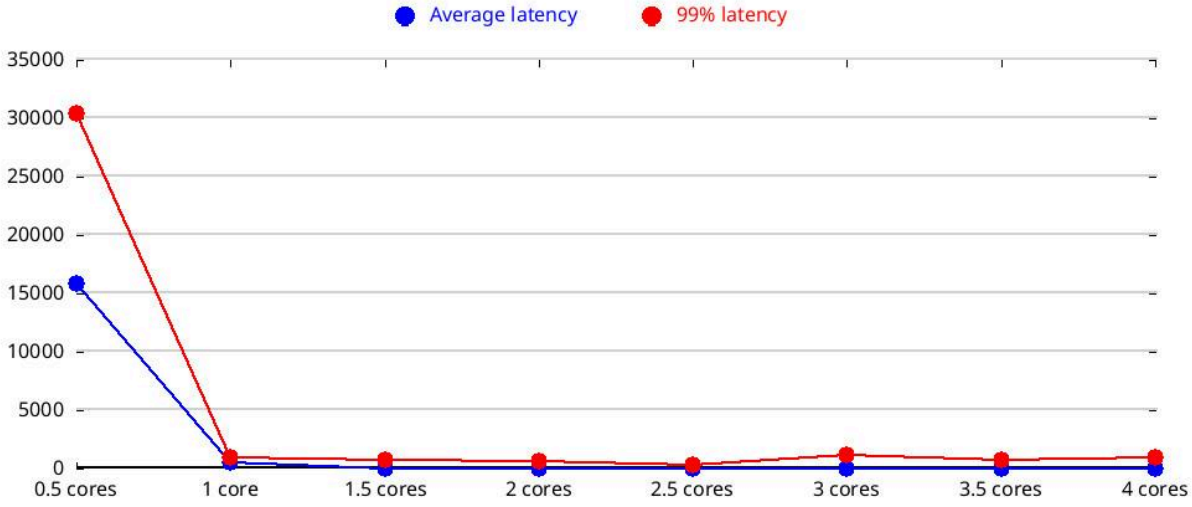
Kafka latency with 10 Jobs

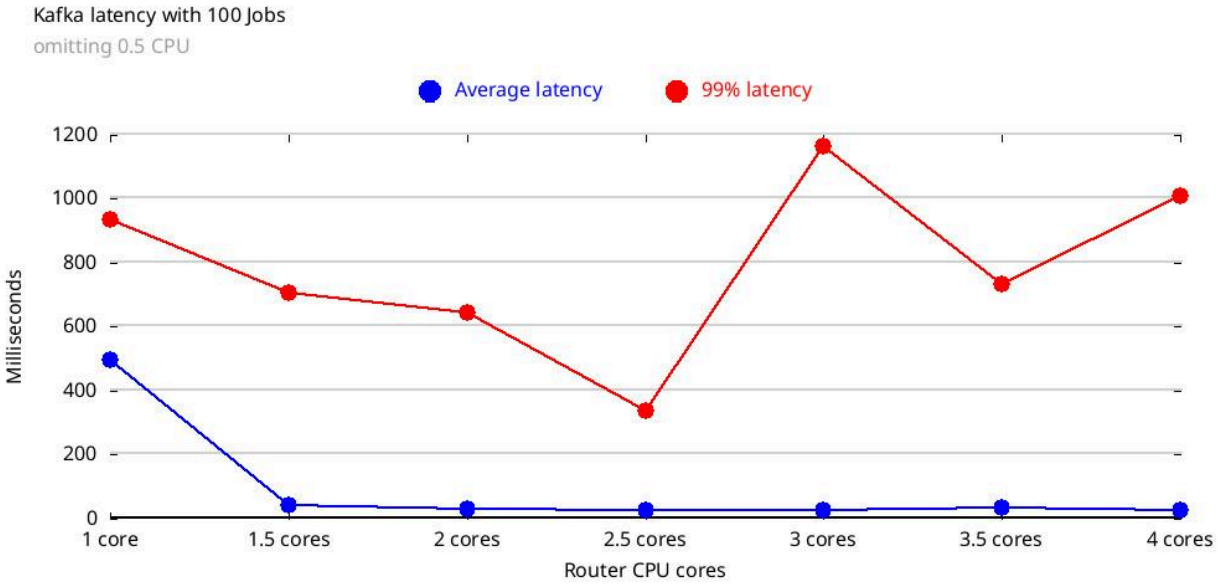
omitting 0.5 CPU



Kafka latency with 100 Jobs

including 0.5 CPU





Recommendation

Router CPU

The primary factor to consider when scaling Skupper for your workload is router CPU. (Note that due to the nature of cluster ingress and connection routing, it's important to focus on scaling the router vertically, not horizontally.)

Two CPU cores (2,000 millicores) per router is a good starting point. It includes some headroom and provides low latencies for a large set of workloads.

If the peak throughput required by your workload is low, it is possible to achieve satisfactory latencies with less router CPU.

Some workloads are sensitive to network latency. In these cases, the overhead introduced by the router can limit the achievable throughput. This is when CPU amounts higher than two cores per router may be required.

On the flip side, some workloads are tolerant of network latency. In these cases, one core or less may be sufficient.

These benchmark results aren't the last word. They depend on the specifics of our test environment. To get a better idea of how Skupper performs in your environment, you can run these benchmarks yourself.

Router memory

Router memory use scales with the number of open connections. In general, a good starting point is 4G.

Memory	**Concurrent open connections**
512M	8,192
1G	16,384
2G	32,768
4G	**65,536**
8G	131,072
16G	262,144
32G	524,288
64G	104,8576

Benchdog

Benchdog is a suite of containerized client-server benchmark tools. It's what we used to produce these results.

These tools are intended for you to use directly, so you can better characterize Skupper performance in your environment. See the GitHub project for source code and instructions.

https://github.com/skupperproject/benchdog

Appendix: Worker node details

```
Linux kube-ck07bf6w067b1v78gb40-jross1-default-00000415
4.18.0-477.21.1.el8_8.x86_64 #1 SMP Thu Jul 20 08:38:27 EDT 2023 x86_64
x86_64 x86_64 GNU/Linux
```

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 85
model name      : Intel Xeon Processor (Cascadelake)
stepping        : 6
microcode       : 0x1
cpu MHz         : 2394.326
cache size      : 16384 KB
physical id     : 0
siblings        : 4
core id         : 0
cpu cores       : 2
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm
constant_tsc rep_good nopl xtopology cpuid tsc_known_freq pni pclmulqdq vmx
ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes
xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch cpuid_fault
invpcid_single ssbd ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi
flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms
invpcid avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw
avx512vl xsaveopt xsavec xgetbv1 xsaves arat pku ospke avx512_vnni md_clear
arch_capabilities
bugs            : spectre_v1 spectre_v2 spec_store_bypass swapgs
mmio_stale_data retbleed eibrs_pbrsb
bogomips        : 4788.65
clflush size    : 64
cache_alignment : 64
address sizes   : 40 bits physical, 48 bits virtual
power management:

processor       : 1
vendor_id       : GenuineIntel
cpu family      : 6
model           : 85
model name      : Intel Xeon Processor (Cascadelake)
```

```
stepping        : 6
microcode       : 0x1
cpu MHz         : 2394.326
cache size      : 16384 KB
physical id     : 0
siblings        : 4
core id         : 0
cpu cores       : 2
apicid          : 1
initial apicid  : 1
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm
constant_tsc rep_good nopl xtopology cpuid tsc_known_freq pni pclmulqdq vmx
ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes
xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch cpuid_fault
invpcid_single ssbd ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi
flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms
invpcid avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw
avx512vl xsaveopt xsavec xgetbv1 xsaves arat pku ospke avx512_vnni md_clear
arch_capabilities
bugs            : spectre_v1 spectre_v2 spec_store_bypass swapgs
mmio_stale_data retbleed eibrs_pbrsb
bogomips        : 4788.65
clflush size    : 64
cache_alignment : 64
address sizes   : 40 bits physical, 48 bits virtual
power management:

processor       : 2
vendor_id       : GenuineIntel
cpu family      : 6
model           : 85
model name      : Intel Xeon Processor (Cascadelake)
stepping        : 6
microcode       : 0x1
cpu MHz         : 2394.326
cache size      : 16384 KB
physical id     : 0
siblings        : 4
```

```
core id         : 1
cpu cores       : 2
apicid          : 2
initial apicid  : 2
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm
constant_tsc rep_good nopl xtopology cpuid tsc_known_freq pni pclmulqdq vmx
ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes
xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch cpuid_fault
invpcid_single ssbd ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi
flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms
invpcid avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw
avx512vl xsaveopt xsavec xgetbv1 xsaves arat pku ospke avx512_vnni md_clear
arch_capabilities
bugs            : spectre_v1 spectre_v2 spec_store_bypass swapgs
mmio_stale_data retbleed eibrs_pbrsb
bogomips        : 4788.65
clflush size    : 64
cache_alignment : 64
address sizes   : 40 bits physical, 48 bits virtual
power management:

processor       : 3
vendor_id       : GenuineIntel
cpu family      : 6
model           : 85
model name      : Intel Xeon Processor (Cascadelake)
stepping        : 6
microcode       : 0x1
cpu MHz         : 2394.326
cache size      : 16384 KB
physical id     : 0
siblings        : 4
core id         : 1
cpu cores       : 2
apicid          : 3
initial apicid  : 3
fpu             : yes
fpu_exception   : yes
```

```
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm
constant_tsc rep_good nopl xtopology cpuid tsc_known_freq pni pclmulqdq vmx
ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes
xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch cpuid_fault
invpcid_single ssbd ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi
flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms
invpcid avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw
avx512vl xsaveopt xsavec xgetbv1 xsaves arat pku ospke avx512_vnni md_clear
arch_capabilities
bugs            : spectre_v1 spectre_v2 spec_store_bypass swapgs
mmio_stale_data retbleed eibrs_pbrsb
bogomips        : 4788.65
clflush size    : 64
cache_alignment : 64
address sizes   : 40 bits physical, 48 bits virtual
power management:

processor       : 4
vendor_id       : GenuineIntel
cpu family      : 6
model           : 85
model name      : Intel Xeon Processor (Cascadelake)
stepping        : 6
microcode       : 0x1
cpu MHz         : 2394.326
cache size      : 16384 KB
physical id     : 1
siblings        : 4
core id         : 0
cpu cores       : 2
apicid          : 4
initial apicid  : 4
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm
constant_tsc rep_good nopl xtopology cpuid tsc_known_freq pni pclmulqdq vmx
ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes
```

```
xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch cpuid_fault
invpcid_single ssbd ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi
flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms
invpcid avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw
avx512vl xsaveopt xsavec xgetbv1 xsaves arat pku ospke avx512_vnni md_clear
arch_capabilities
bugs            : spectre_v1 spectre_v2 spec_store_bypass swapgs
mmio_stale_data retbleed eibrs_pbrsb
bogomips        : 4788.65
clflush size    : 64
cache_alignment : 64
address sizes   : 40 bits physical, 48 bits virtual
power management:

processor       : 5
vendor_id       : GenuineIntel
cpu family      : 6
model           : 85
model name      : Intel Xeon Processor (Cascadelake)
stepping        : 6
microcode       : 0x1
cpu MHz         : 2394.326
cache size      : 16384 KB
physical id     : 1
siblings        : 4
core id         : 0
cpu cores       : 2
apicid          : 5
initial apicid  : 5
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm
constant_tsc rep_good nopl xtopology cpuid tsc_known_freq pni pclmulqdq vmx
ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes
xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch cpuid_fault
invpcid_single ssbd ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi
flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms
invpcid avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw
avx512vl xsaveopt xsavec xgetbv1 xsaves arat pku ospke avx512_vnni md_clear
arch_capabilities
```

```
bugs            : spectre_v1 spectre_v2 spec_store_bypass swapgs
mmio_stale_data retbleed eibrs_pbrsb
bogomips        : 4788.65
clflush size    : 64
cache_alignment : 64
address sizes   : 40 bits physical, 48 bits virtual
power management:

processor       : 6
vendor_id       : GenuineIntel
cpu family      : 6
model           : 85
model name      : Intel Xeon Processor (Cascadelake)
stepping        : 6
microcode       : 0x1
cpu MHz         : 2394.326
cache size      : 16384 KB
physical id     : 1
siblings        : 4
core id         : 1
cpu cores       : 2
apicid          : 6
initial apicid  : 6
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm
constant_tsc rep_good nopl xtopology cpuid tsc_known_freq pni pclmulqdq vmx
ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes
xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch cpuid_fault
invpcid_single ssbd ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi
flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms
invpcid avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw
avx512vl xsaveopt xsavec xgetbv1 xsaves arat pku ospke avx512_vnni md_clear
arch_capabilities
bugs            : spectre_v1 spectre_v2 spec_store_bypass swapgs
mmio_stale_data retbleed eibrs_pbrsb
bogomips        : 4788.65
clflush size    : 64
cache_alignment : 64
address sizes   : 40 bits physical, 48 bits virtual
```

```
power management:

processor       : 7
vendor_id       : GenuineIntel
cpu family      : 6
model           : 85
model name      : Intel Xeon Processor (Cascadelake)
stepping        : 6
microcode       : 0x1
cpu MHz         : 2394.326
cache size      : 16384 KB
physical id     : 1
siblings        : 4
core id         : 1
cpu cores       : 2
apicid          : 7
initial apicid  : 7
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm
constant_tsc rep_good nopl xtopology cpuid tsc_known_freq pni pclmulqdq vmx
ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes
xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch cpuid_fault
invpcid_single ssbd ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi
flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms
invpcid avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw
avx512vl xsaveopt xsavec xgetbv1 xsaves arat pku ospke avx512_vnni md_clear
arch_capabilities
bugs            : spectre_v1 spectre_v2 spec_store_bypass swapgs
mmio_stale_data retbleed eibrs_pbrsb
bogomips        : 4788.65
clflush size    : 64
cache_alignment : 64
address sizes   : 40 bits physical, 48 bits virtual
power management:
```