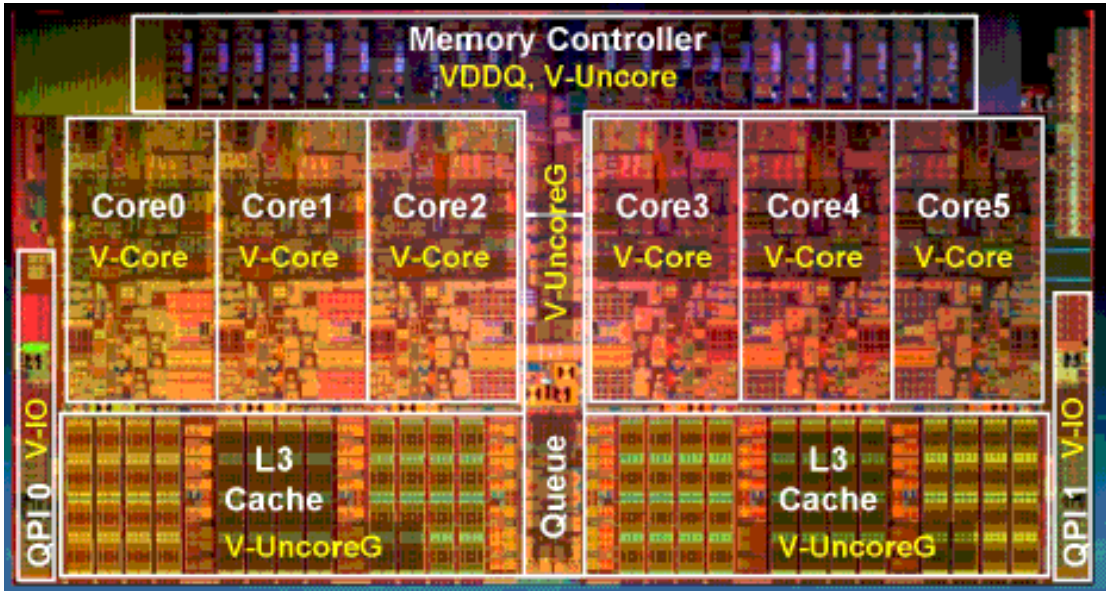
# Datacenter application interference




- CMPs (popular in datacenters) offer increased throughput and reduced power consumption
- They also increase resource sharing between applications, which can result in negative interference.

# Resource contention is well studied

… at least on single machines.

3 main methods:

(1) Gladiator style match-ups

(2) Static analysis to predict application resource usage

(3) Measure benchmark resource usage; apply to live applications

# New methodology for understanding datacenter interference is needed.

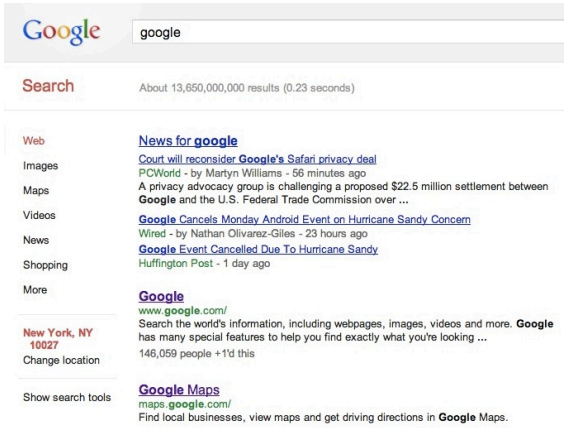
One that can handle complexities of a datacenter:

- ✓ (10s of) thousands of applications
- ✓ real user inputs
- ✓ production hardware
- ✓ financially feasible
- ✓ low overhead

Hardware counter measurements of live applications.

# Our contributions

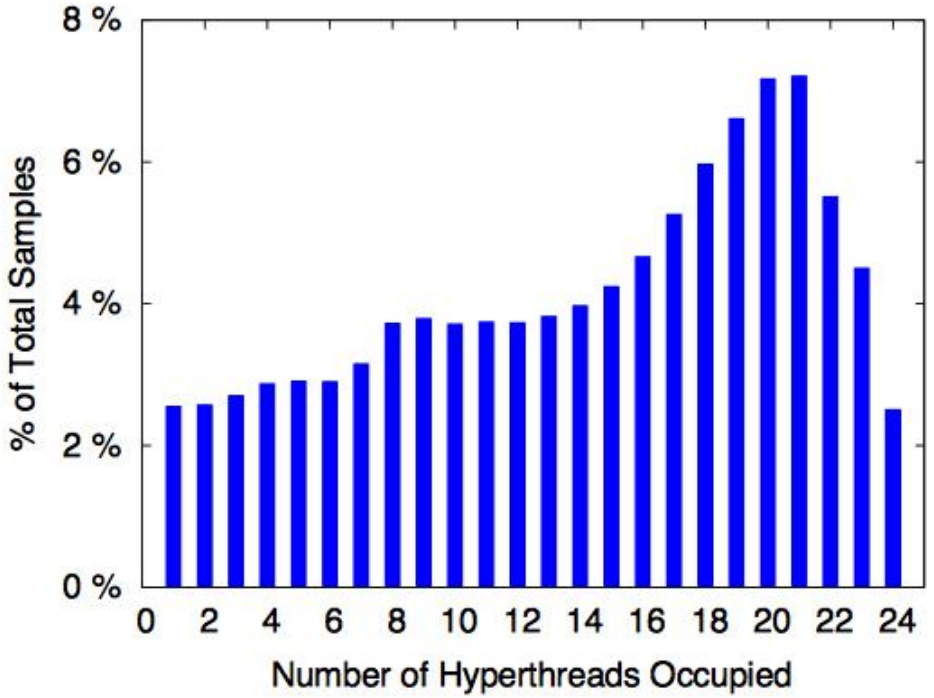
1. ID complexities in datacenters
2. New measurement methodology
3. First large-scale study of measured interference on live datacenter applications.



# Complexities of understanding application interference in a datacenter

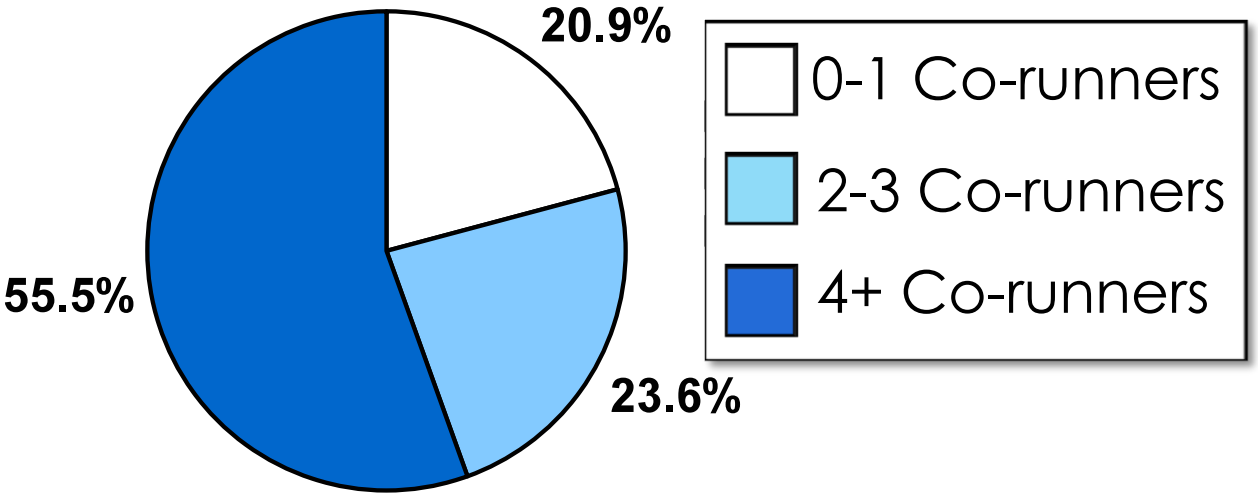
# Large chips and high core utilizations

Profiling 1000 12-core, 24-hyperthread Google servers running production workloads revealed the average machine had >14/24 HW threads in use.



# Heterogeneous application mixes

Often applications have more than one co-runner on a machine.



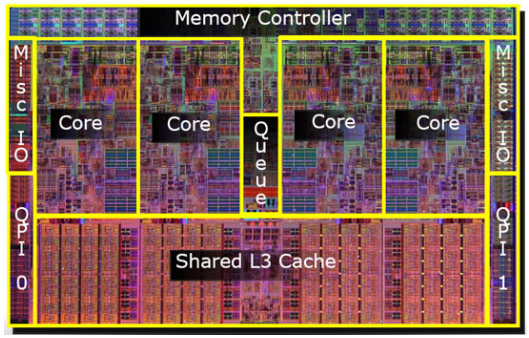

**Observed max of 19 unique co-runner threads (out of 24 HW threads).**

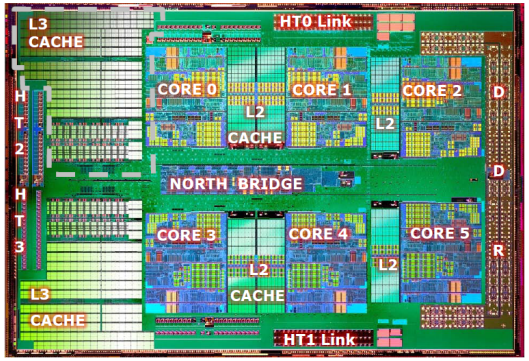
# Application complexities

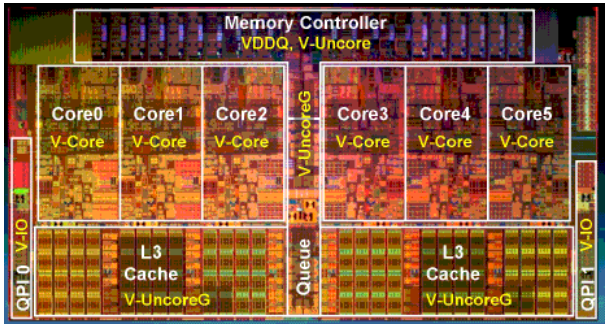
- Fuzzy definitions
- Varying and sometimes unpredictable inputs
- Unknown optimal performance

# Hardware & Economic Complexities

- Varying micro-arch platforms



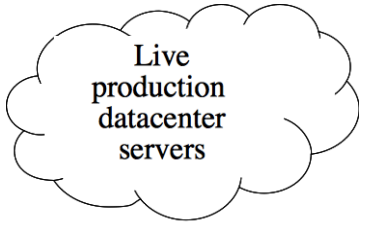




- Necessity for low overhead = limited measurement capabilities
- Corporate policies

# Measurement methodology

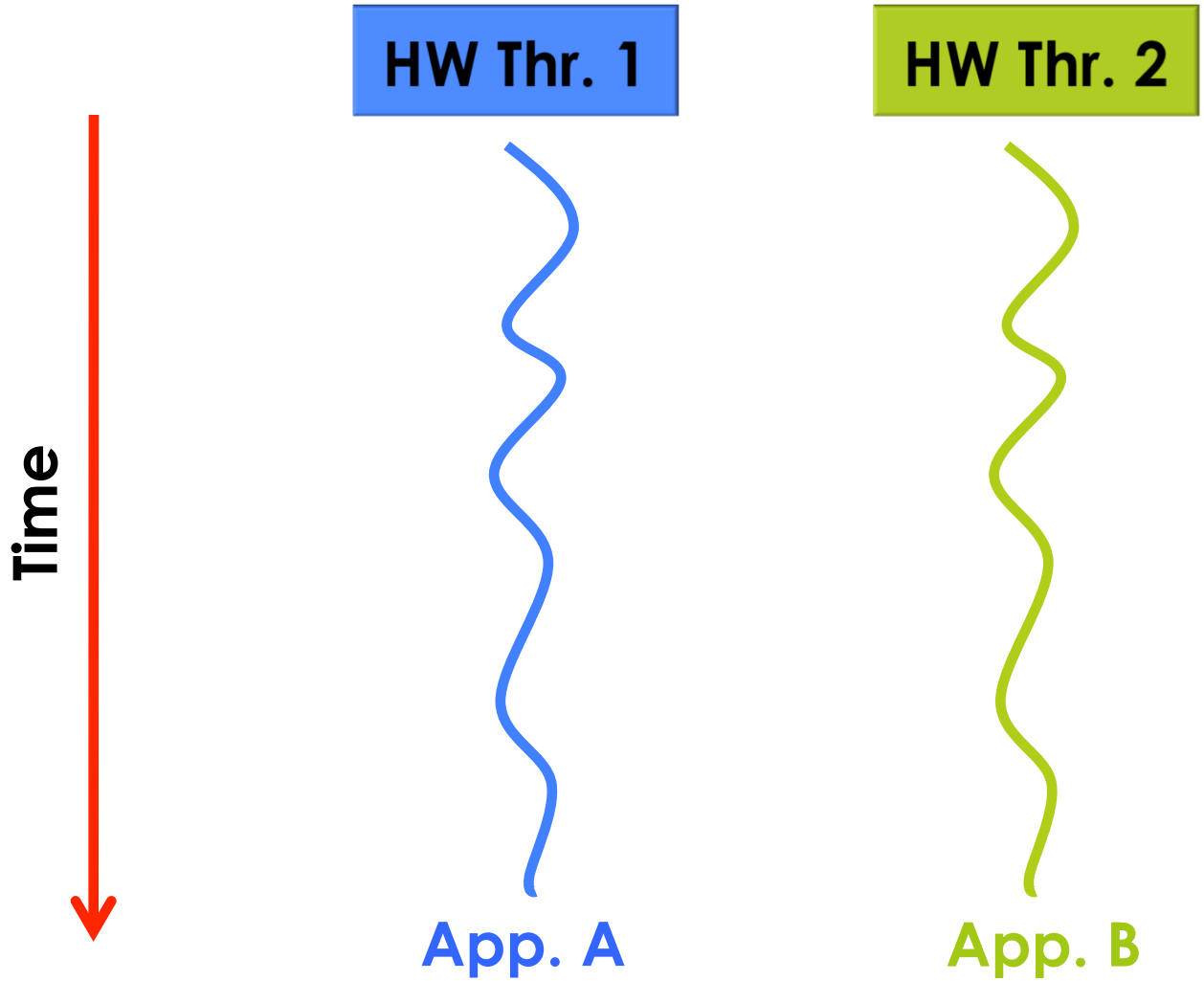
# Measurement Methodology



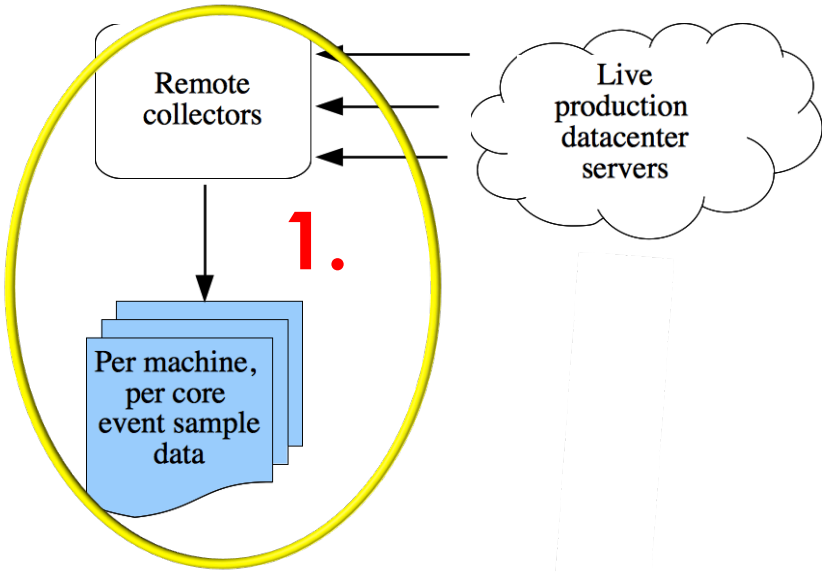

The goal:

A generic methodology to collect application interference data on live production datacenter servers

# Measurement Methodology

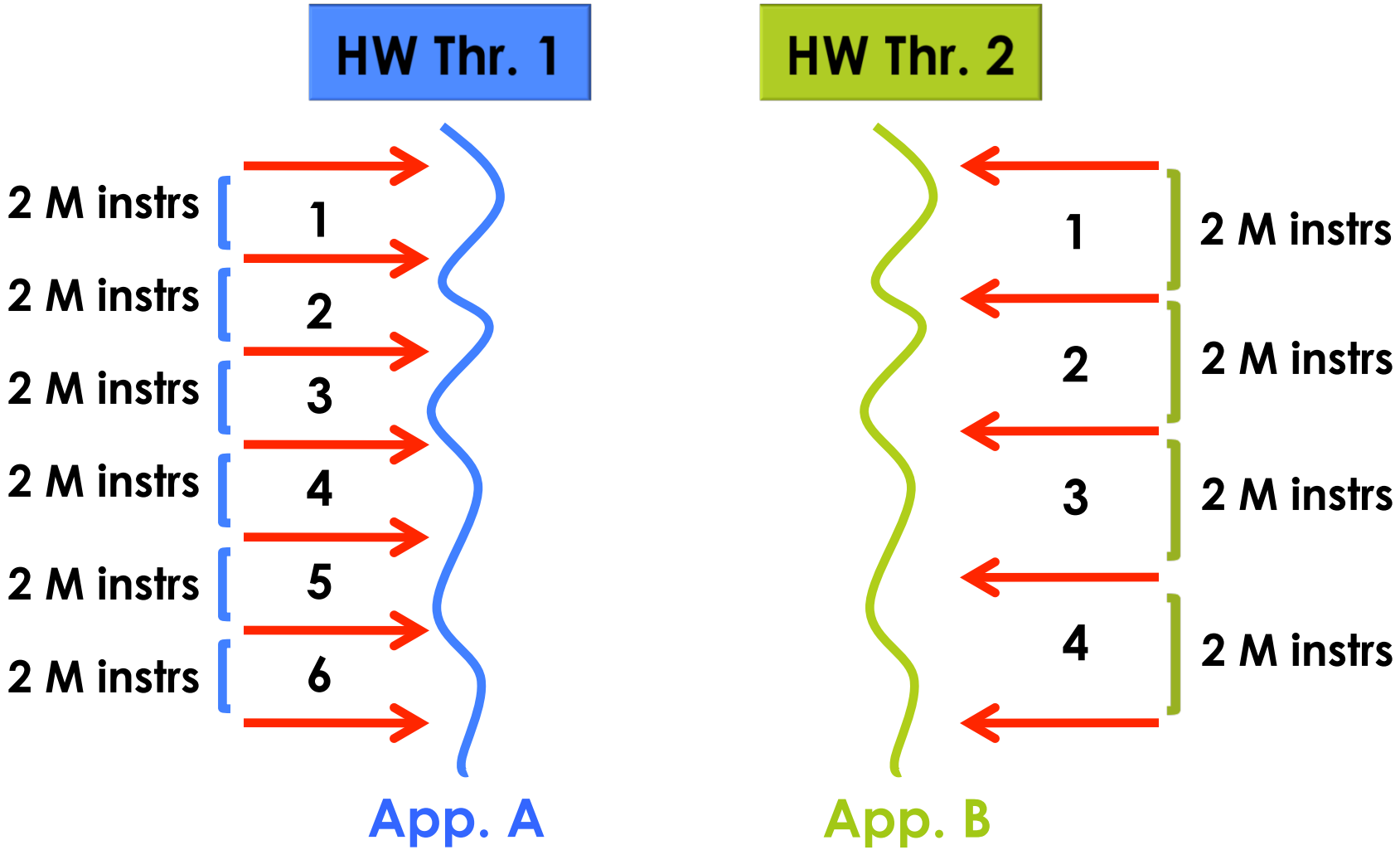


# Measurement Methodology

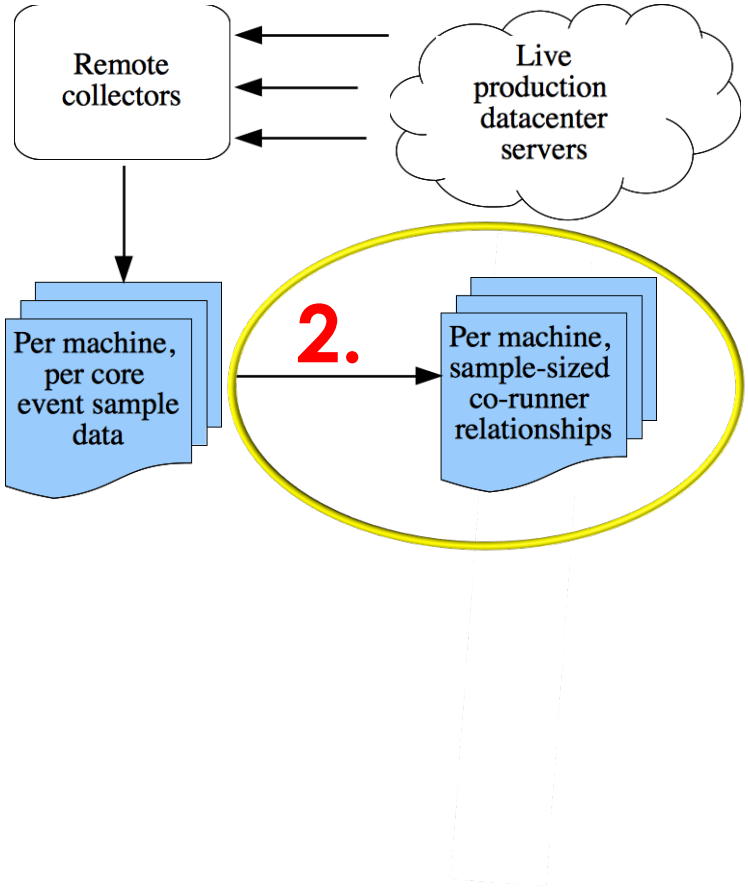



**1.** Use sample-based monitoring to collect per machine per core *event* (HW counter) sample data.

# Measurement Methodology

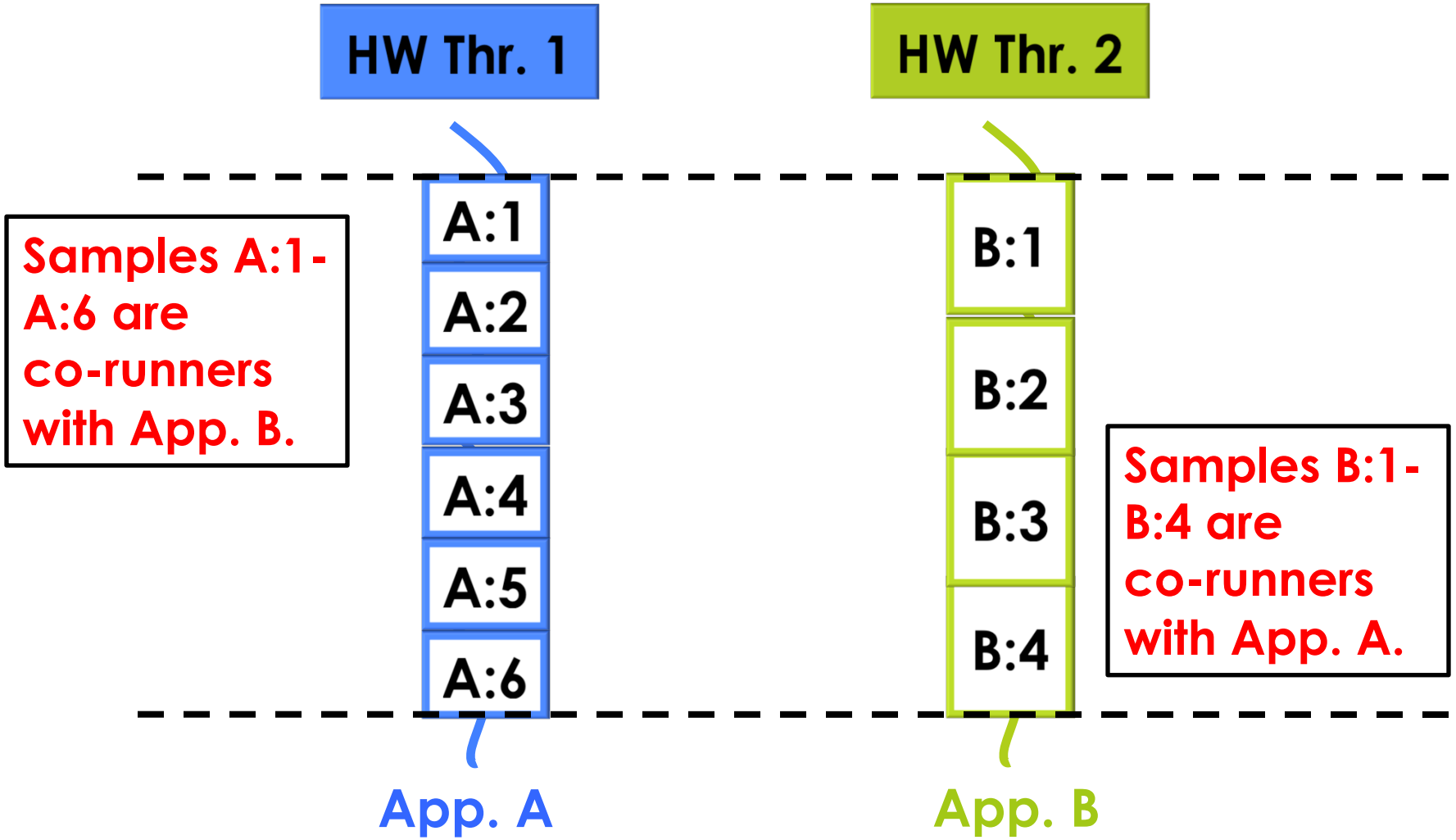


# Measurement Methodology

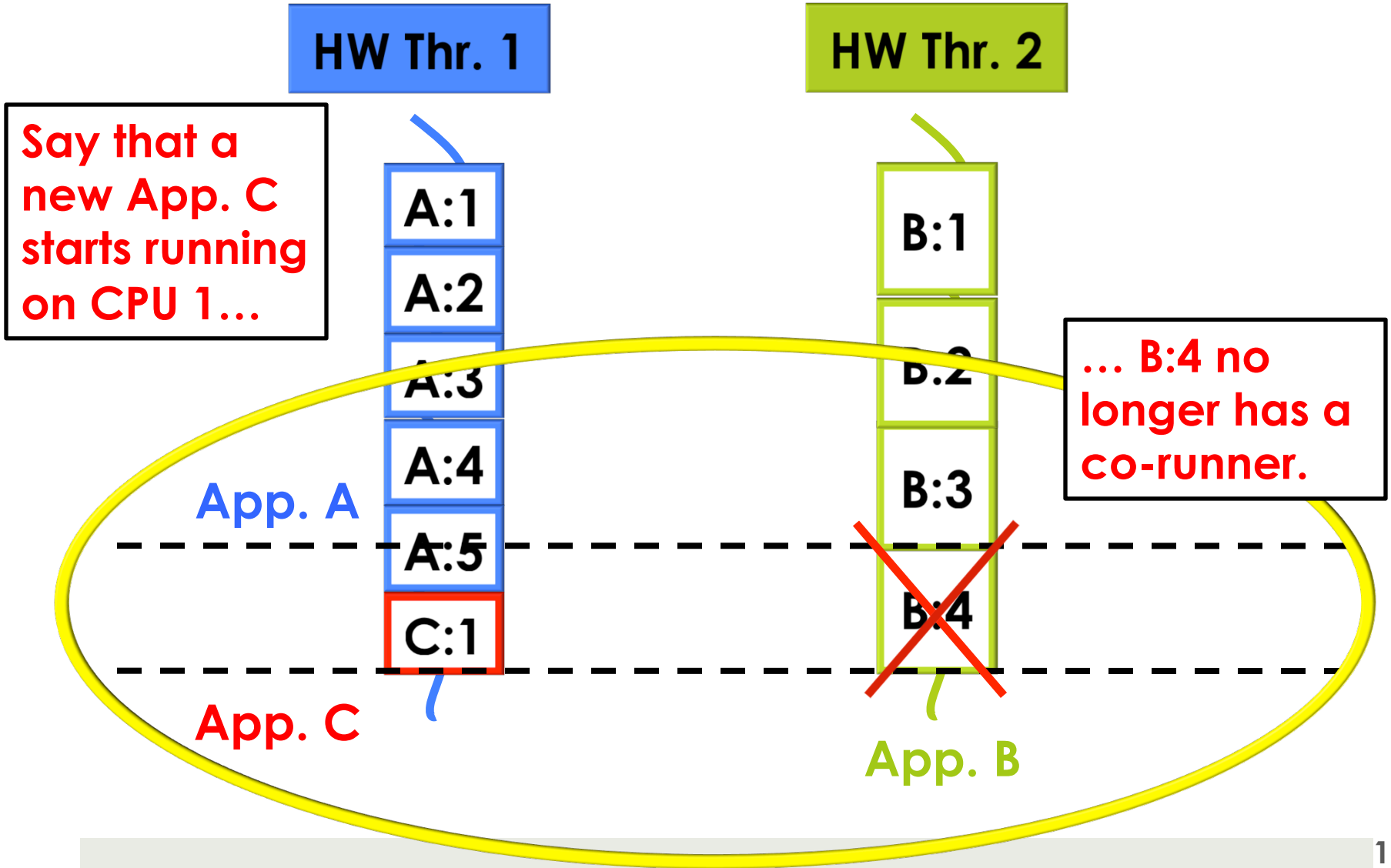



**2.** Identify sample sized *co-runner relationships*…

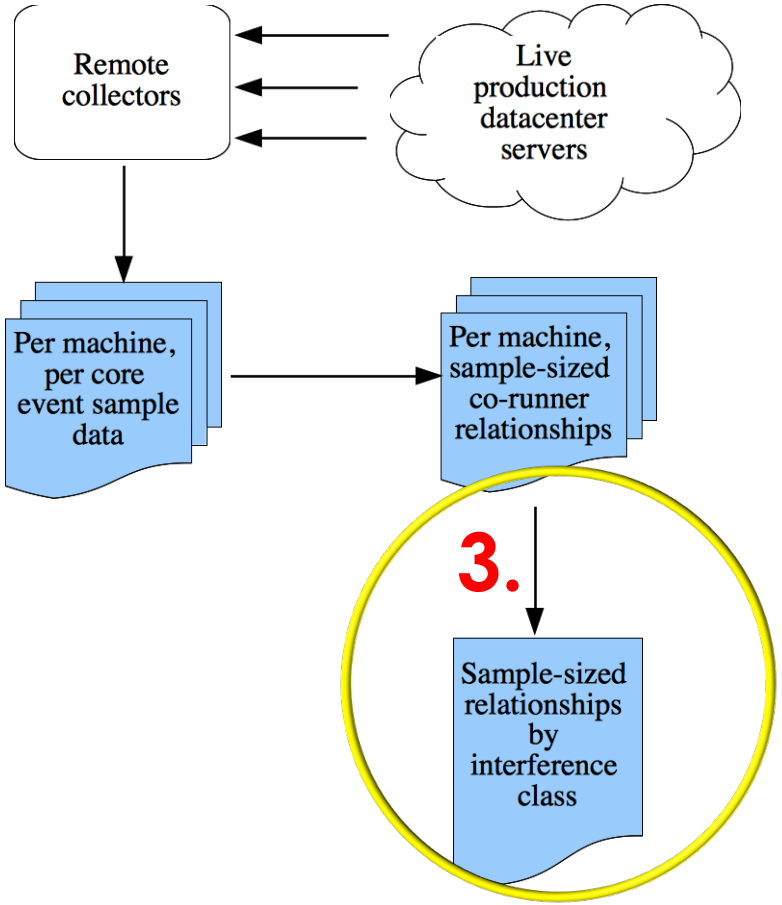
# Measurement Methodology



# Measurement Methodology

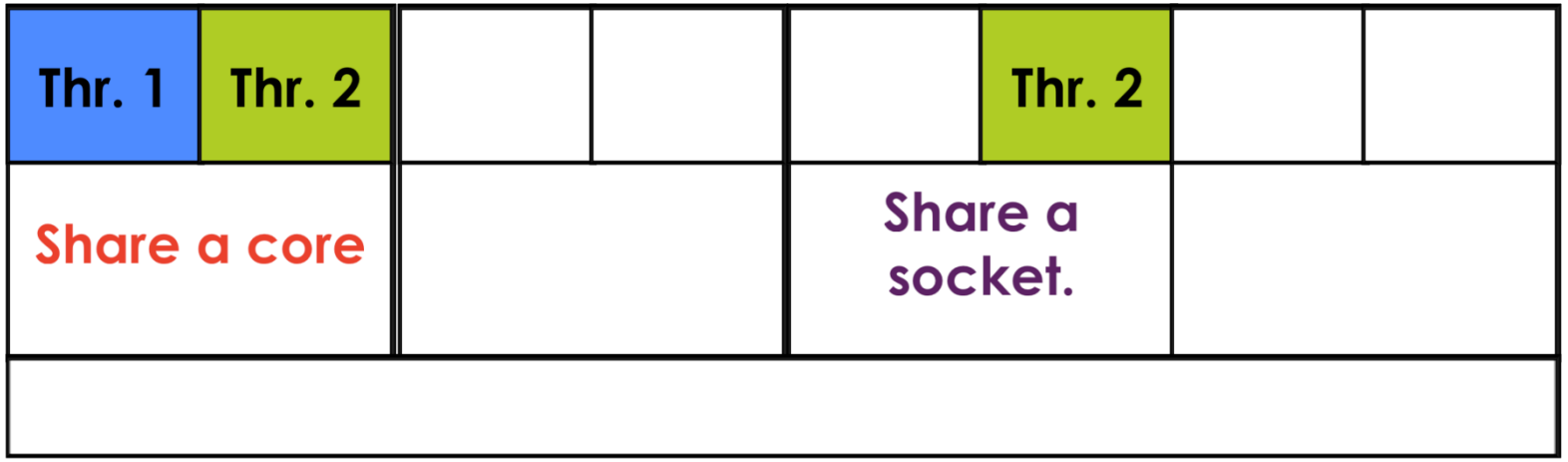


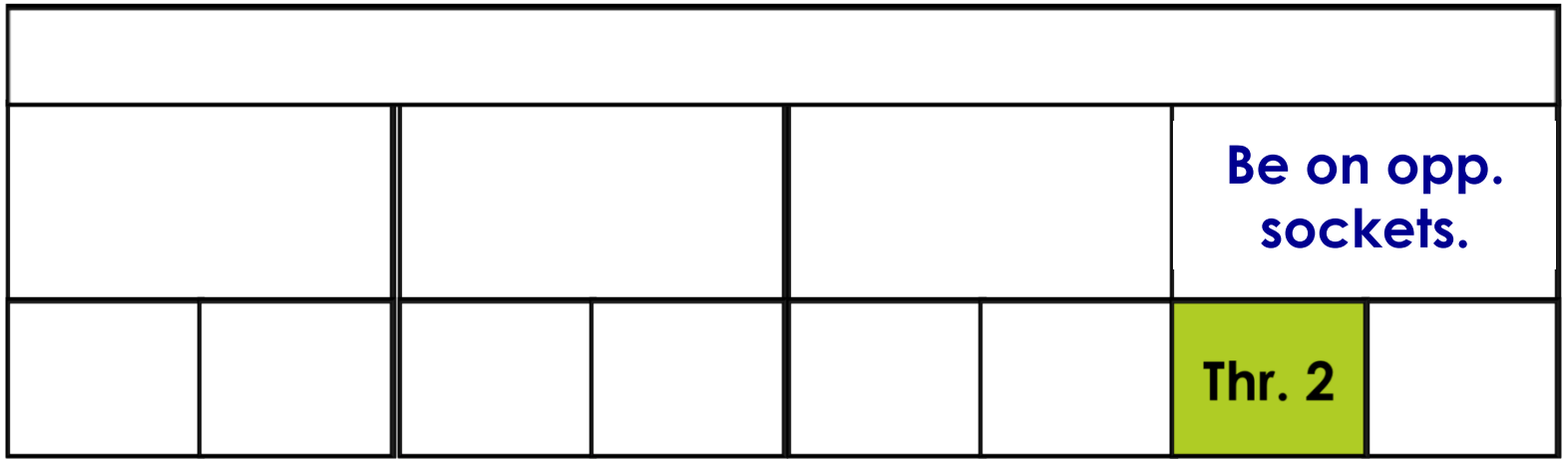
# Measurement Methodology



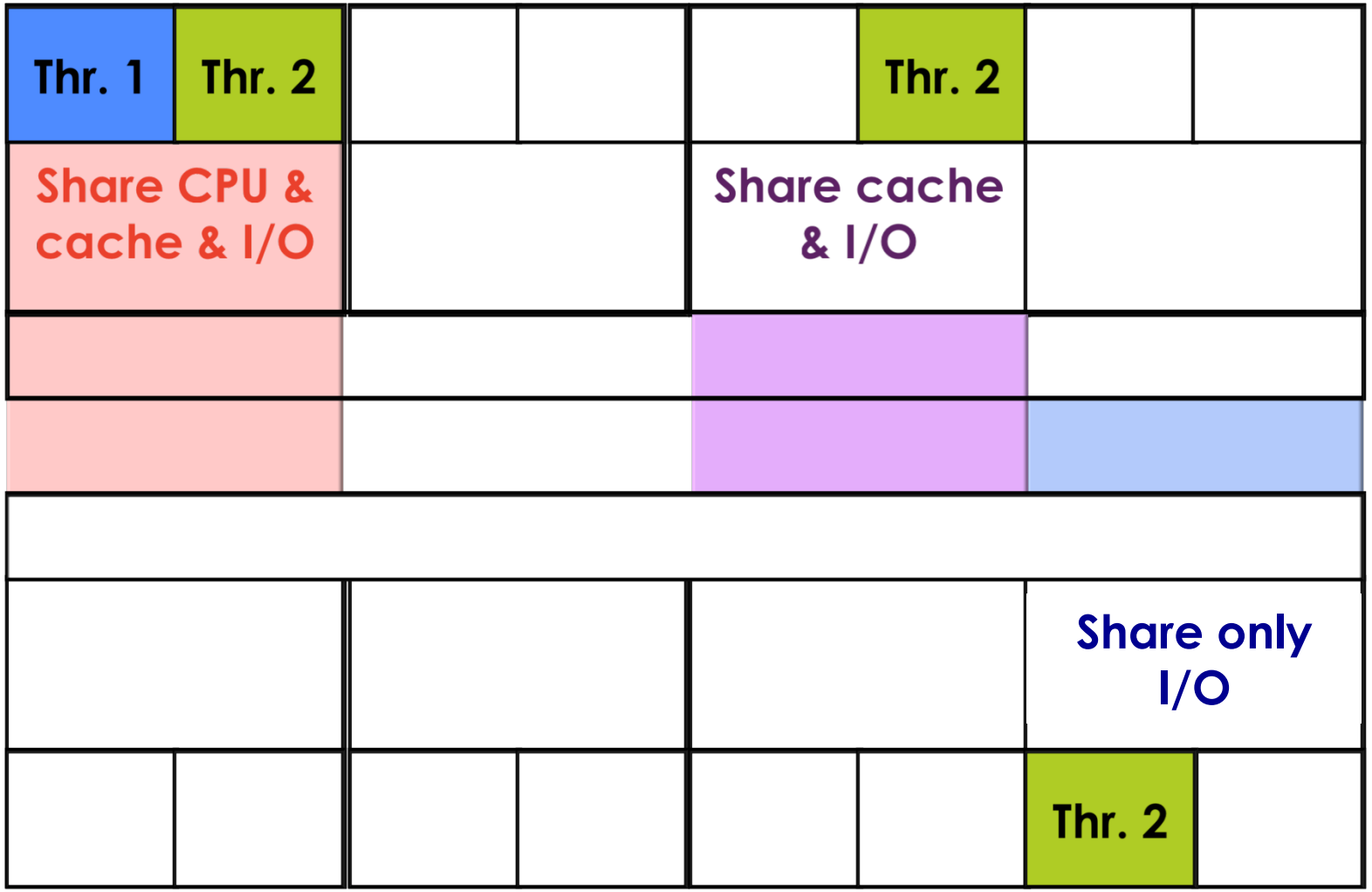

**3.** Filter relationships by arch. independent *interference classes*…

# Measurement Methodology

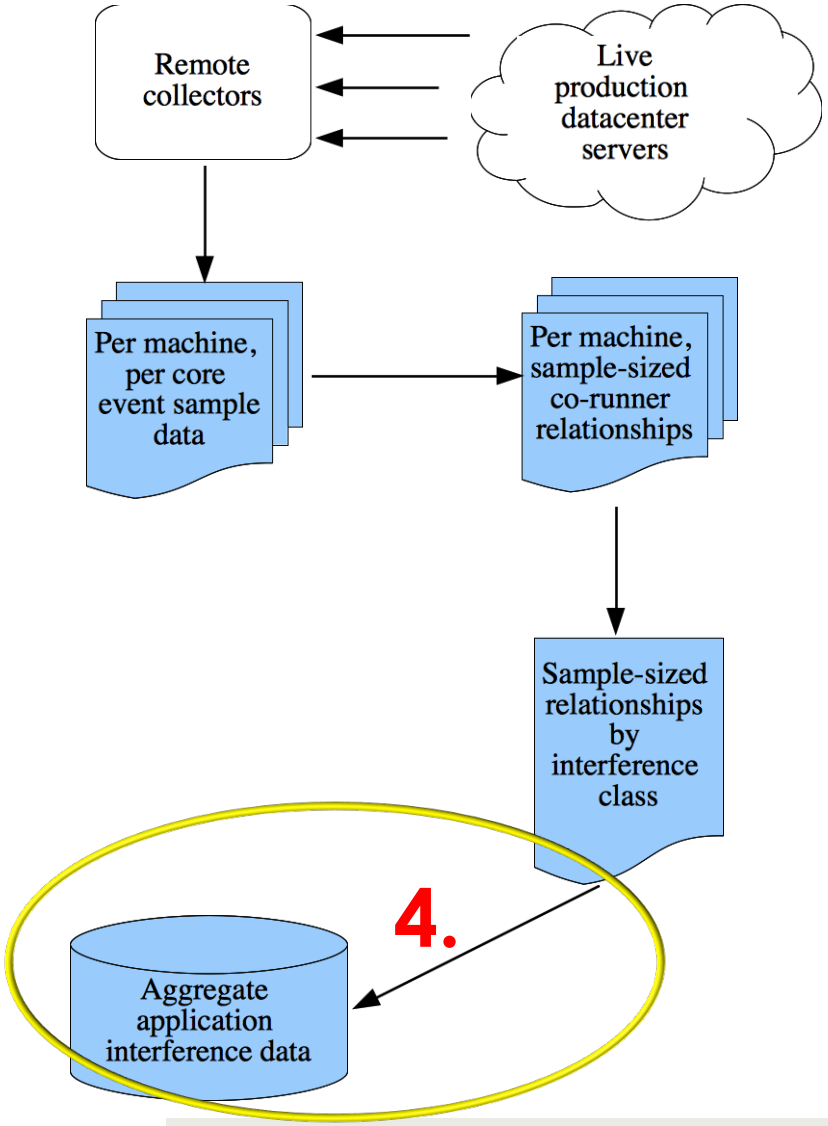


# Measurement Methodology



# Measurement Methodology



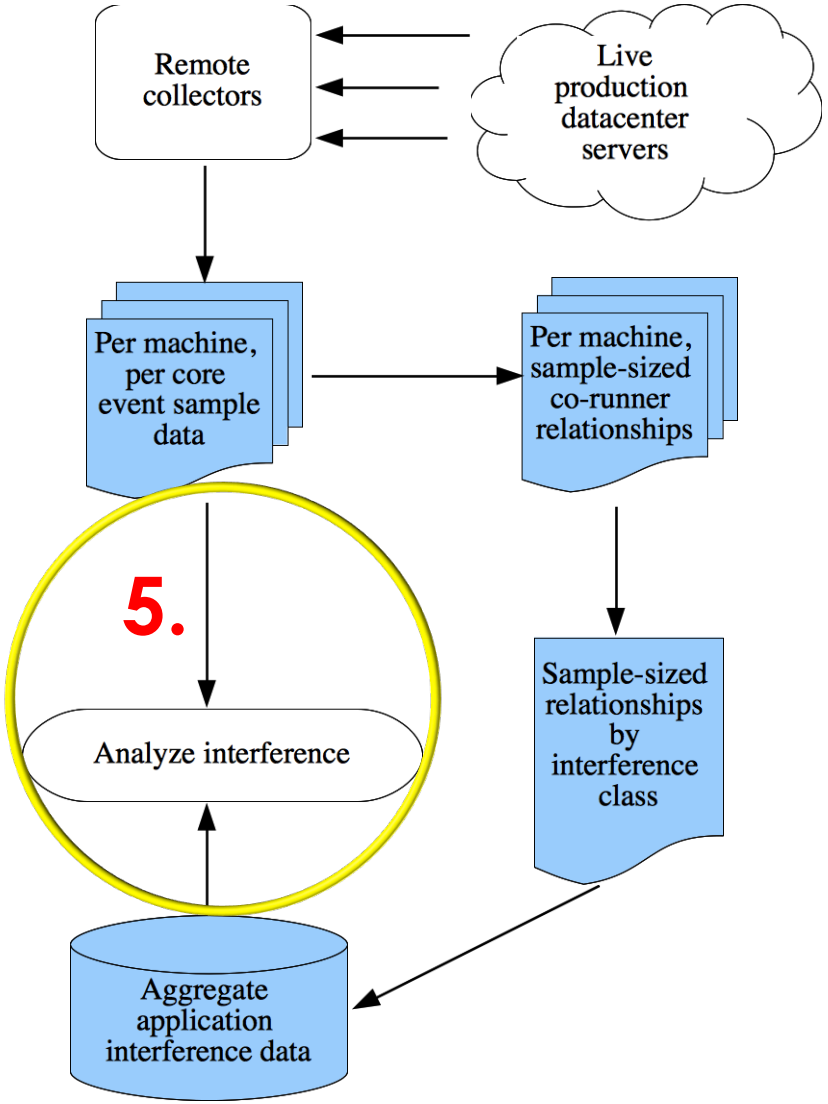

**4.** Aggregate equivalent *co-schedules*.

# Measurement Methodology

For example:

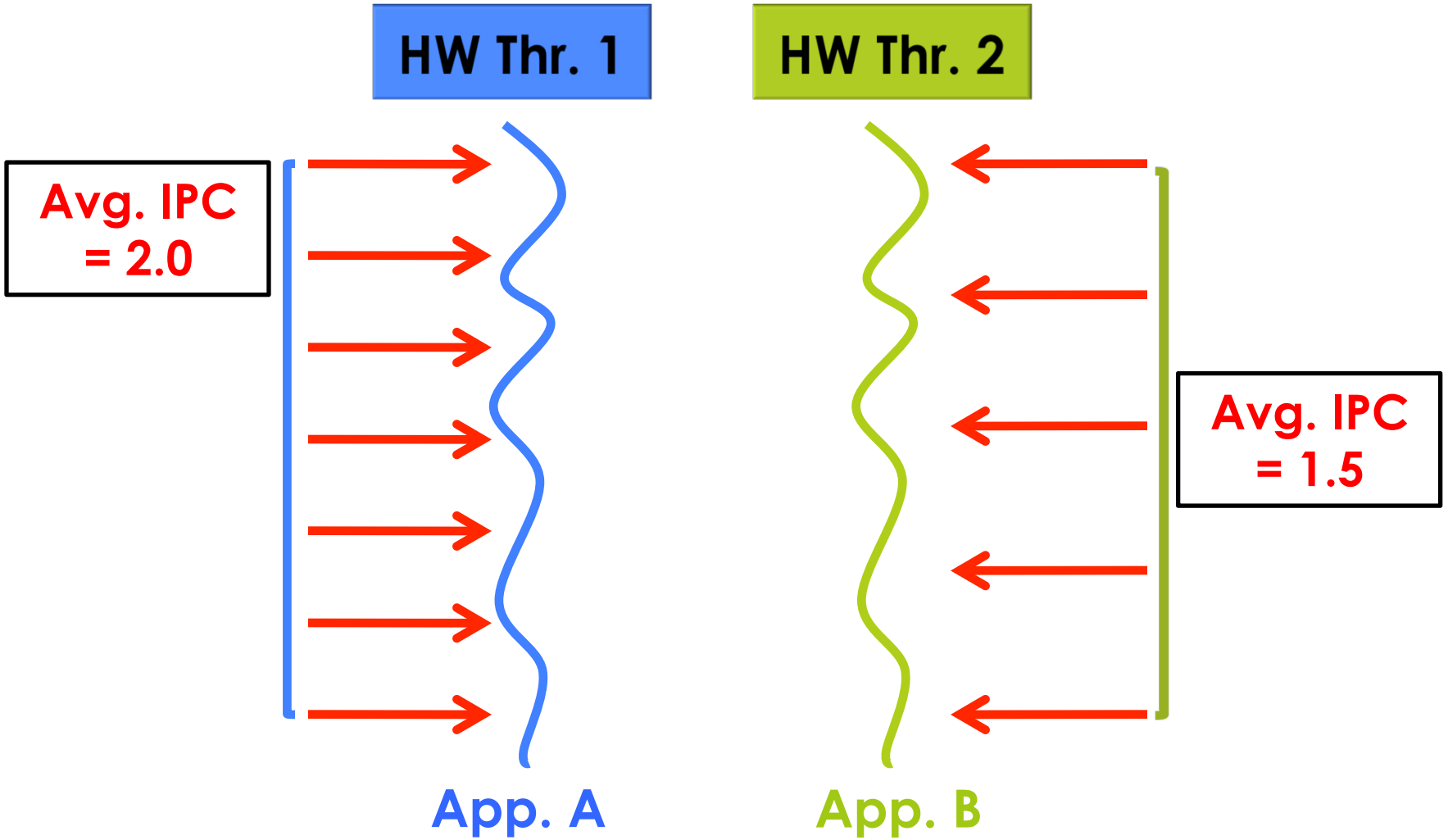
- Aggregate all the samples of App. A that have App. B as a shared core co-runner.
- Aggregate all samples of App. A that have App. B as a shared core co-runner *and* App. C as a shared socket co-runner.

# Measurement Methodology




**5.** Finally, calculate *statistical indicators* (means, medians) to get a midpoint performance for app. interference comparisons

# Measurement Methodology



# Applying the measurement methodology at Google.

# Applying the Methodology @ Google

## Experiment Details:

| Event | Instrs ➔ IPC |
|---|---|
| Sampling period | 2.5 Million |
| Number of machines* | 1000 |

* All had Intel Westmere chips (24 hyperthreads, 12 cores), matching clock speed, RAM, O/S

## Method:

1. Collect samples

# Applying the Methodology @ Google

## Experiment Details:

| Event | Instrs ➔ IPC |
|---|---|
| Sampling period | 2.5 Million |
| Number of machines* | 1000 |

* All had Intel Westmere chips (24 hyperthreads, 12 cores), matching clock speed, RAM, O/S

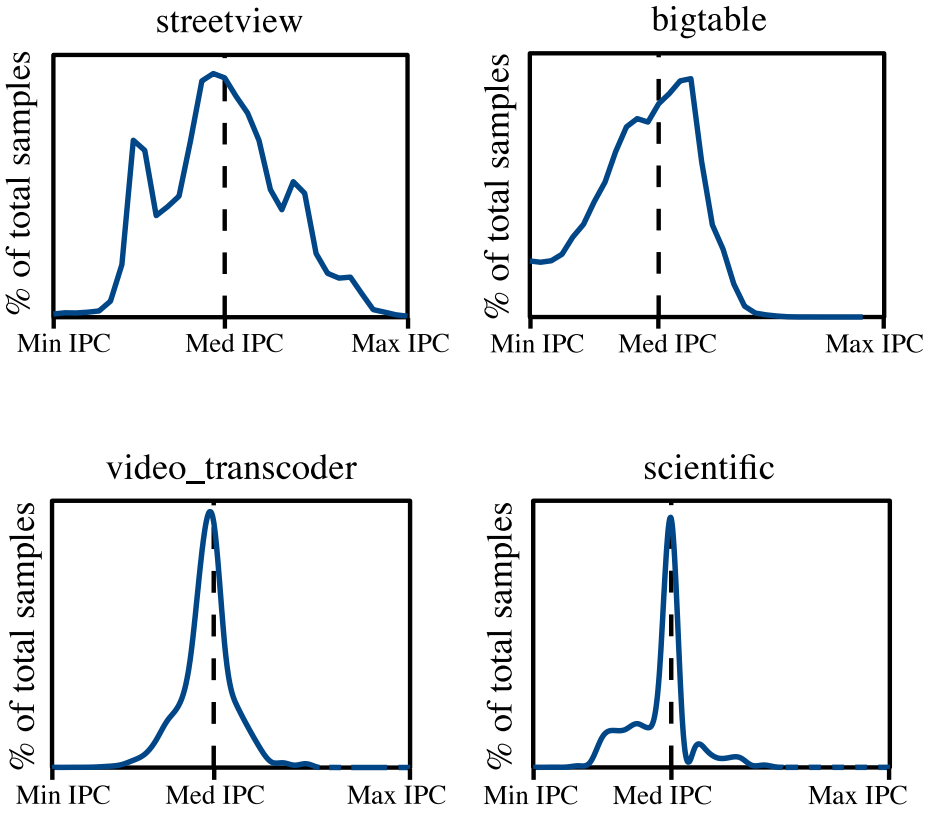
## Collection results:

| Unique binary apps | 1102 |
|---|---|
| Co-runner relationships (top 8 apps) | |
| Avg. shared core rel'ns | 1M (min 2K) |
| Avg. shared socket | 9.5M (min 12K) |
| Avg. opposite socket | 11M (min 14K) |

## Method:

1. Collect samples
2. ID sample size relationships
3. Filter by interference classes

# Applying the Methodology @ Google



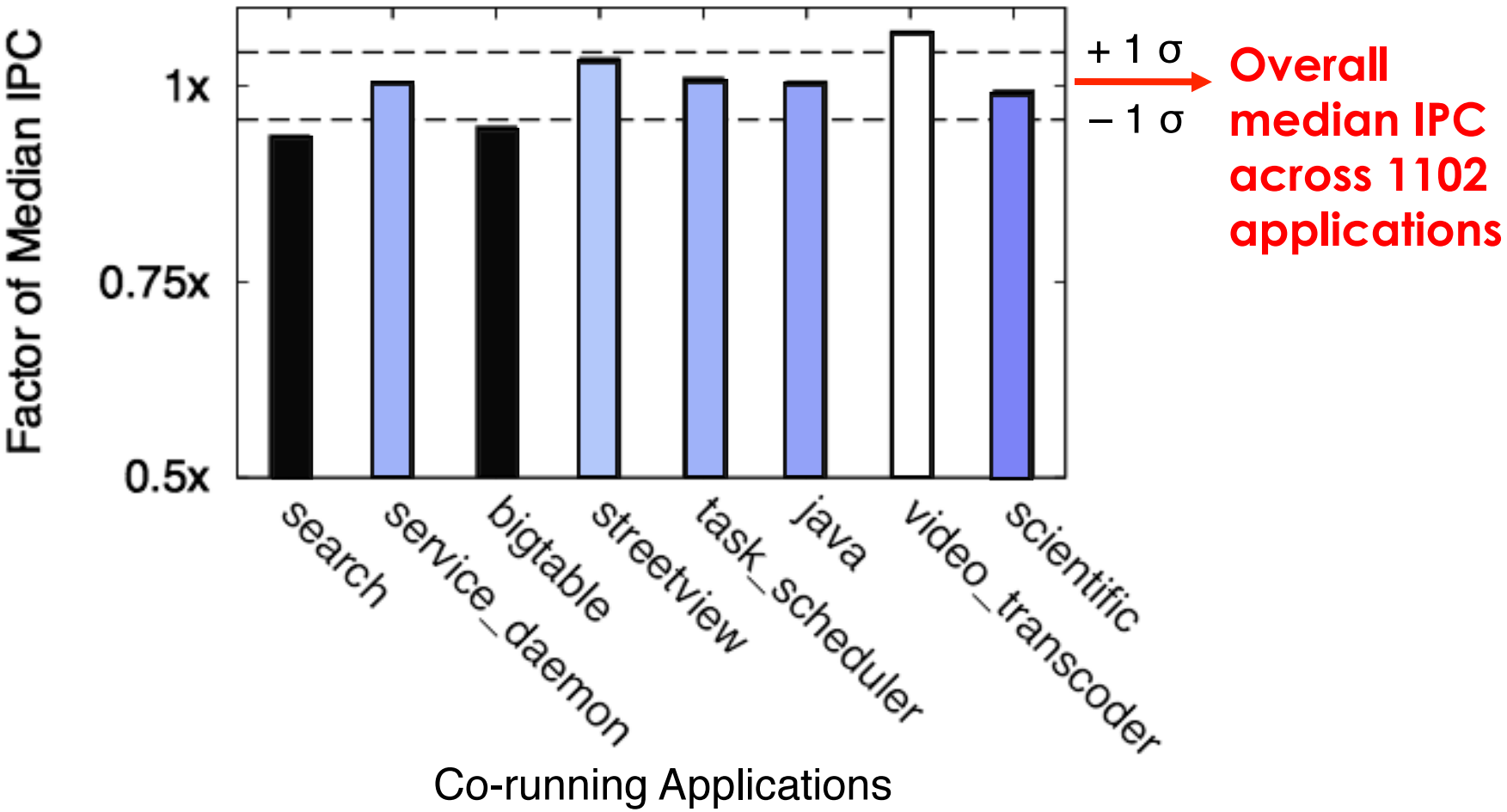

**Method:**

4. Aggregate equiv. schedules

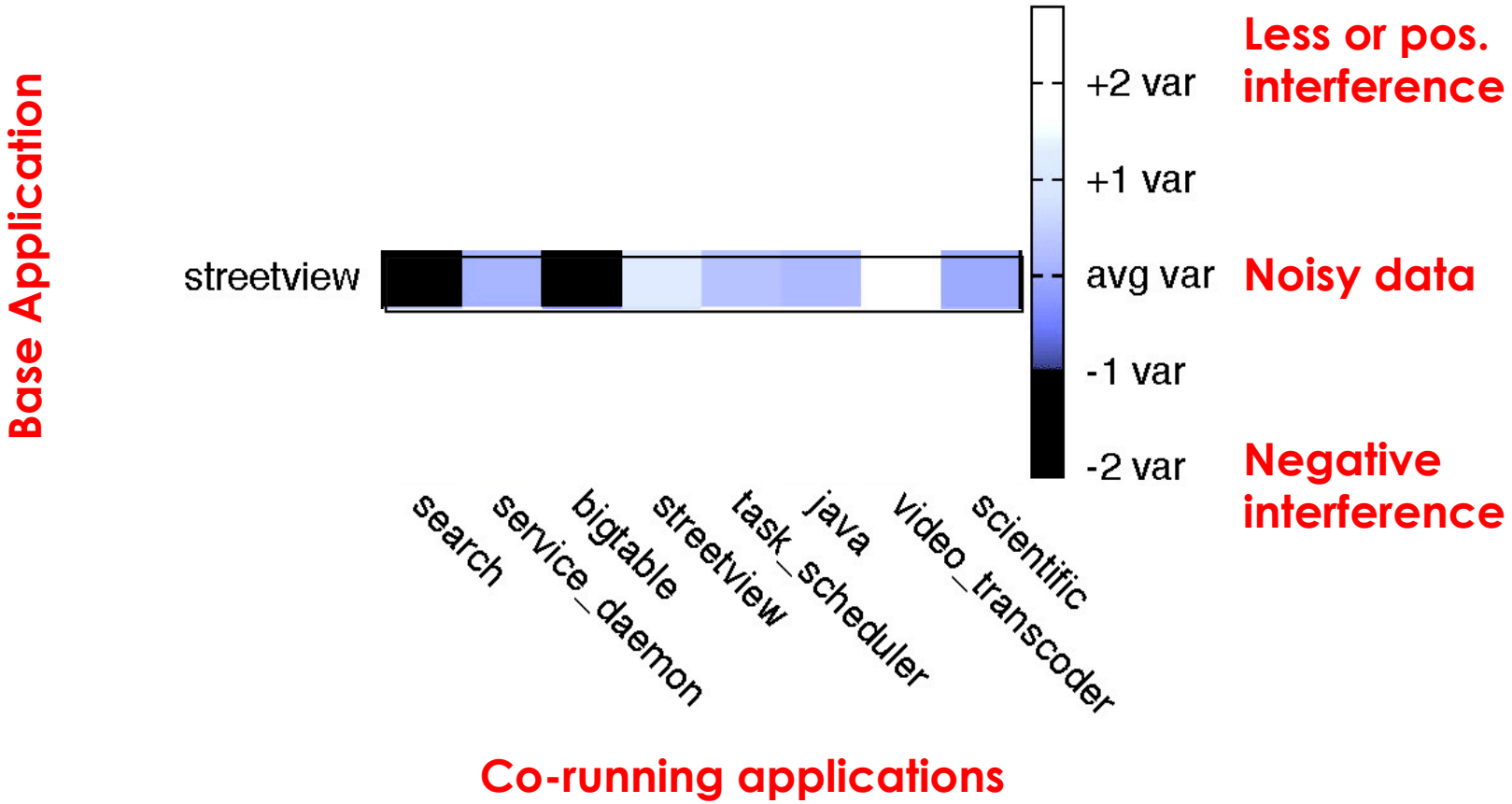
5. Calculate statistical indicators

# Analyze Interference

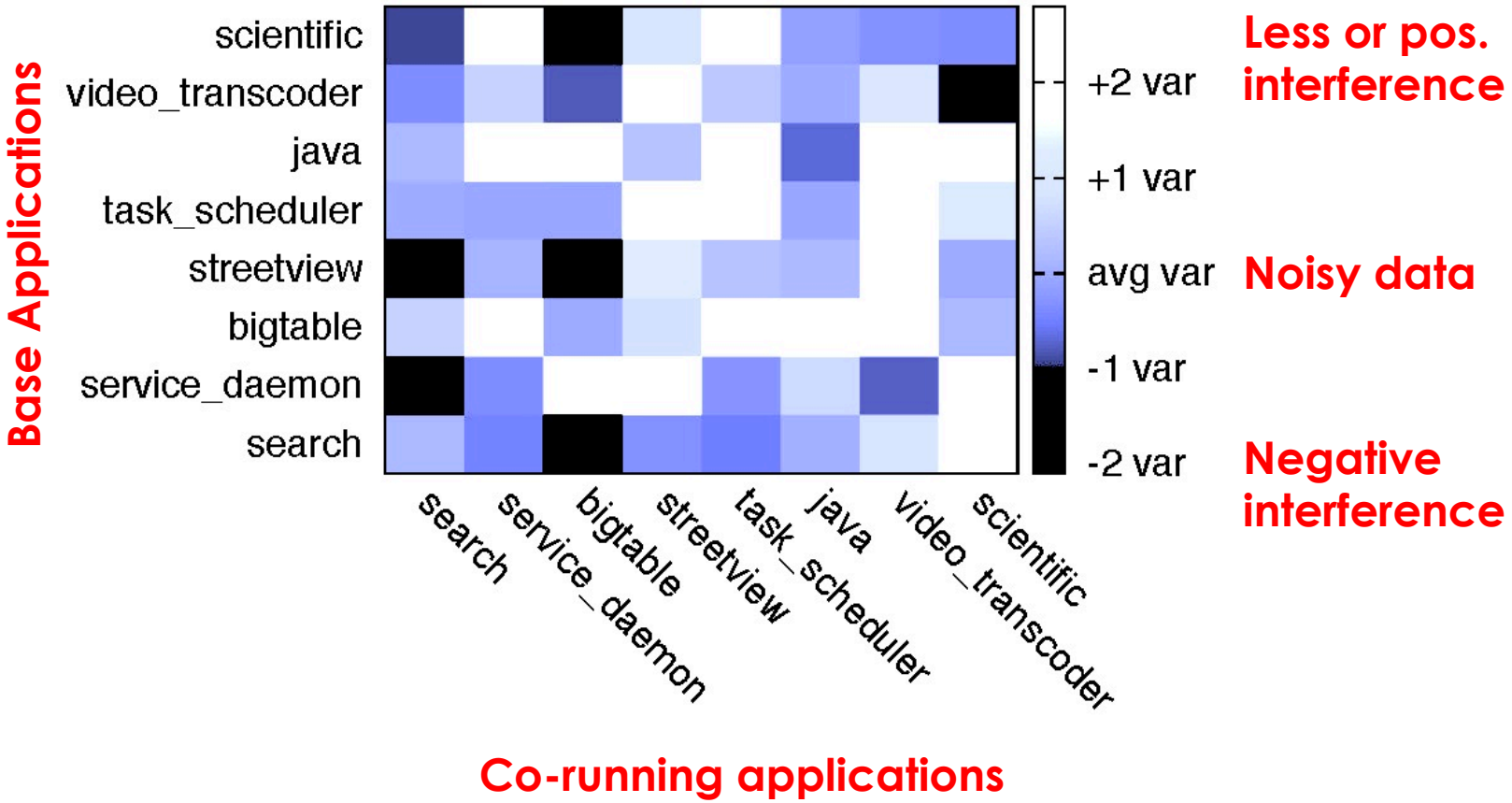
**`streeview`'s IPC changes with top co-runners**



# Beyond noisy interferers (shared core)



# Beyond noisy interferers (shared core)




*** Recall minimum pair has 2K samples; medians across full grid of 1102 apps**

# Performance Strategies

- Restrict negative beyond noisy interferers (or encourage positive interferers as co-runners)

- Isolate sensitive or antagonistic applications

# Takeaways

1. New datacenter application interference studies can use our identified complexities as a check list.
2. Our measurement methodology (verified at Google in 1st large-scale measurements of live datacenter interference), is generally applicable and shows promising initial performance opportunities.

# Questions?

**melanie@cs.columbia.edu**
**http://arcade.cs.columbia.edu/**