Present algorithm for fetching disk utilization with the linux application **iostat:**
ios/iostat_linux.go:

```
    d := cmn.GCO.Get().Periodic.IostatTime
    if err := r.execCmd(d); err != nil {
        return err
    }
...
    go func() {
        // reader loop
        for {
            b, err := r.reader.ReadBytes('\n')
            if r.process == nil {
                return
            }
            if err == io.EOF {
                continue
            } else if err != nil {
                if err = r.retry(2); err != nil {
                    r.stopCh <- err
                    return
                }
            }
            responseCh <- string(b)
        }
    }()

    // main loop
    for {
        select {
        case line := <-responseCh:
            fields := strings.Fields(line)
            if len(fields) < iostatnumdsk {
                continue
            }
            if strings.HasPrefix(fields[0], "Device") {
                if len(r.metricNames) == 0 {
                    r.metricNames = append(r.metricNames, fields[1:]...)
                }
                continue
            }
            device := fields[0]
            if mpath, ok := r.disks2mpath[device]; ok {
                mpathInfo := r.stats.availablePaths[mpath]
                lines[device] = strings.Join(fields, ", ")
                for i := 1; i < len(fields); i++ {
                    name := r.metricNames[i-1]
                    fieldVal, err := strconv.ParseFloat(fields[i], 32)
```

```
					if err != nil {
						continue
					}
					if name == "%util" {
						mpathInfo.SetIOstats(epoch, fs.StatDiskUtil, float32(fieldVal))
					} else if name == "aqu-sz" || name == "avgqu-sz" {
						mpathInfo.SetIOstats(epoch, fs.StatQueueLen, float32(fieldVal))
					}
				}
			}
...
		}
	}
...
func (r *IostatRunner) execCmd(period time.Duration) error {
	if r.process != nil {
		// kill previous process if running - can happen on config change
		if err := r.process.Kill(); err != nil {
			return err
		}
	}

	refreshPeriod := int(period / time.Second)
	cmd := exec.Command("iostat", "-dxm", strconv.Itoa(refreshPeriod)) // the iostat command
	stdout, err := cmd.StdoutPipe()
	r.reader = bufio.NewReader(stdout)
	if err != nil {
		return err
	}
	if err = cmd.Start(); err != nil {
		return err
	}
	r.process = cmd.Process
	return nil
}
```

Proposed algorithm by polling the file **/proc/diskstats**:
ios/iostat_linux.go:

```
	// main loop
	for {
		select {
...
			fetchedDiskStats := GetDiskStats()
			for disk, mpath := range r.disks2mpath {
				stat, ok := fetchedDiskStats[disk]
				if !ok {
					continue
				}
				mpathInfo := r.stats.availablePaths[mpath]

				mpathInfo.SetIOstats(epoch, fs.StatDiskIOms, float32(stat.IOMs))
				if prev, cur := mpathInfo.GetIOstats(fs.StatDiskIOms); prev.Max != 0 {
					msElapsed := d.Nanoseconds() / (1000 * 1000) //convert to Milliseconds
					mpathInfo.SetIOstats(epoch, fs.StatDiskUtil, float32(cur.Max-prev.Max)*100/float32(msElapsed))
				}

				if lc >= lm {
					lines[disk] = stat.ToString()
				}
			}

			if lc >= lm {
				log(lines)
				lines = make(cmn.SimpleKVs, 16)
				lc = 0
			}
		}
	}
```

ios/diskstats_linux.go:

```
type DiskStat struct {
...
}
...
type DiskStats map[string]DiskStat
...
func GetDiskStats() (output DiskStats) {
	output = make(DiskStats)

	file, err := os.Open("/proc/diskstats")
	if err != nil {
		glog.Error(err)
		return
	}

	scanner := bufio.NewScanner(file)

	for scanner.Scan() {
		line := scanner.Text()
		if line == "" {
			continue
		}

		fields := strings.Fields(line)
```

```
        if len(fields) < 14 {
            continue
        }

        deviceName := fields[2]
        output[deviceName] = DiskStat{
...
        }
    }

    return output
}
```

Important factors which need to be considered when replacing the algorithm for fetching disk utilization:

1. Latency
- Since the disk utilization from **iostat** is passed to stdout after it is generated, there is no way to tell how long generating the data took. Thus, only the latency of polling from **/proc/diskstats** itself can be examined. Note that there is no configuration for **iostat** to generate disk util more frequently than every second, so there's no practical difference if the latency for **/proc/diskstats** is lower than a second.

2. Accuracy
- Disk utilization is currently used to determine when to throttle lru. This means that the algorithm replacing **iostat** must produce similar results.

Method

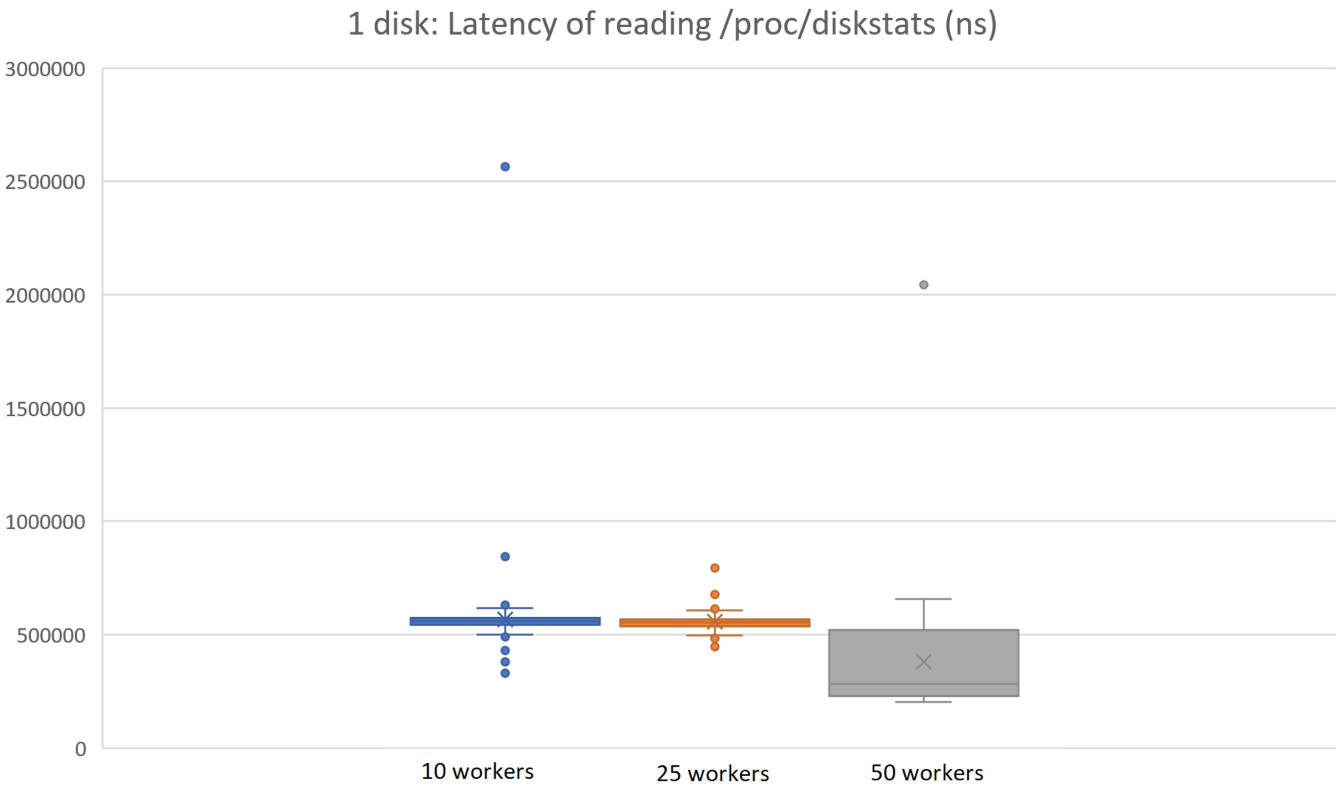
The script is run during all experiments to constantly monitor the disk utilization calculated by both **iostat** and **/proc/diskstats**, along with the latency for **/proc/diskstats**.

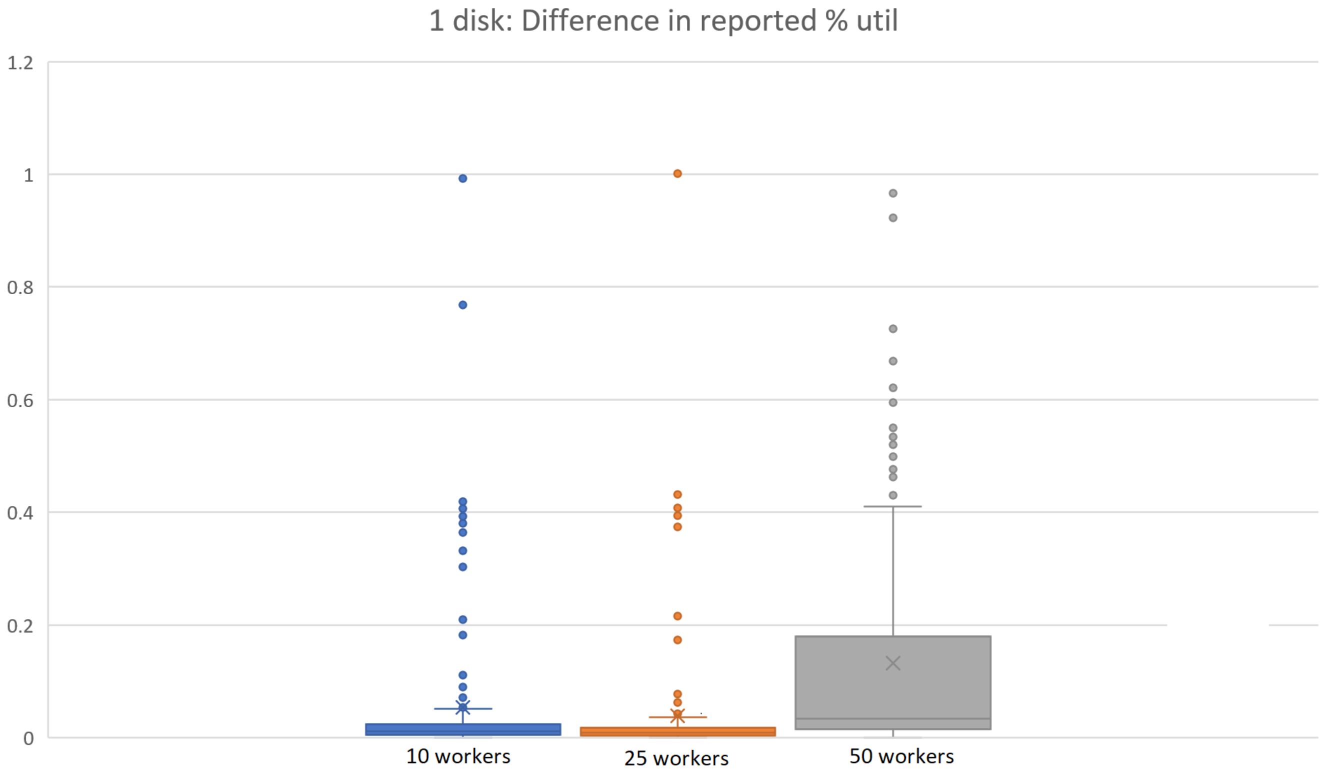
../loadgen/loadgen.sh is run once in every experiment with the fixed arguments: "seconds=200 iobatch=1000 pct_read=75". The number of workers and disks vary by experiment.

The disks are Seagate ST10000NM0096 10TB SAS.

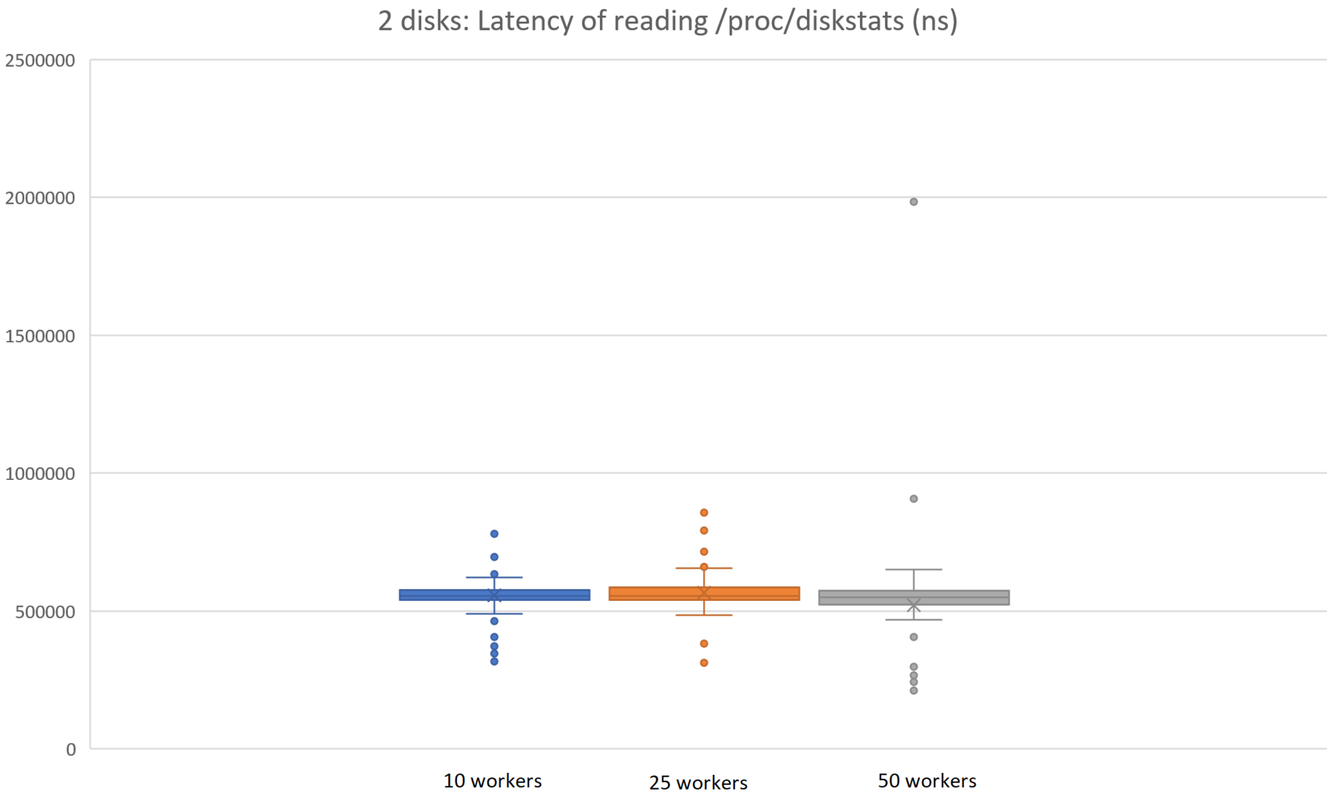
The time it takes to poll /proc/diskstats is recorded, along with the difference in reported percent disk utilization between **iostats** and /**proc/diskstats**. Note that in experiments with multiple disks, the difference in reported percent disk utilization is calculated for each individual disk.

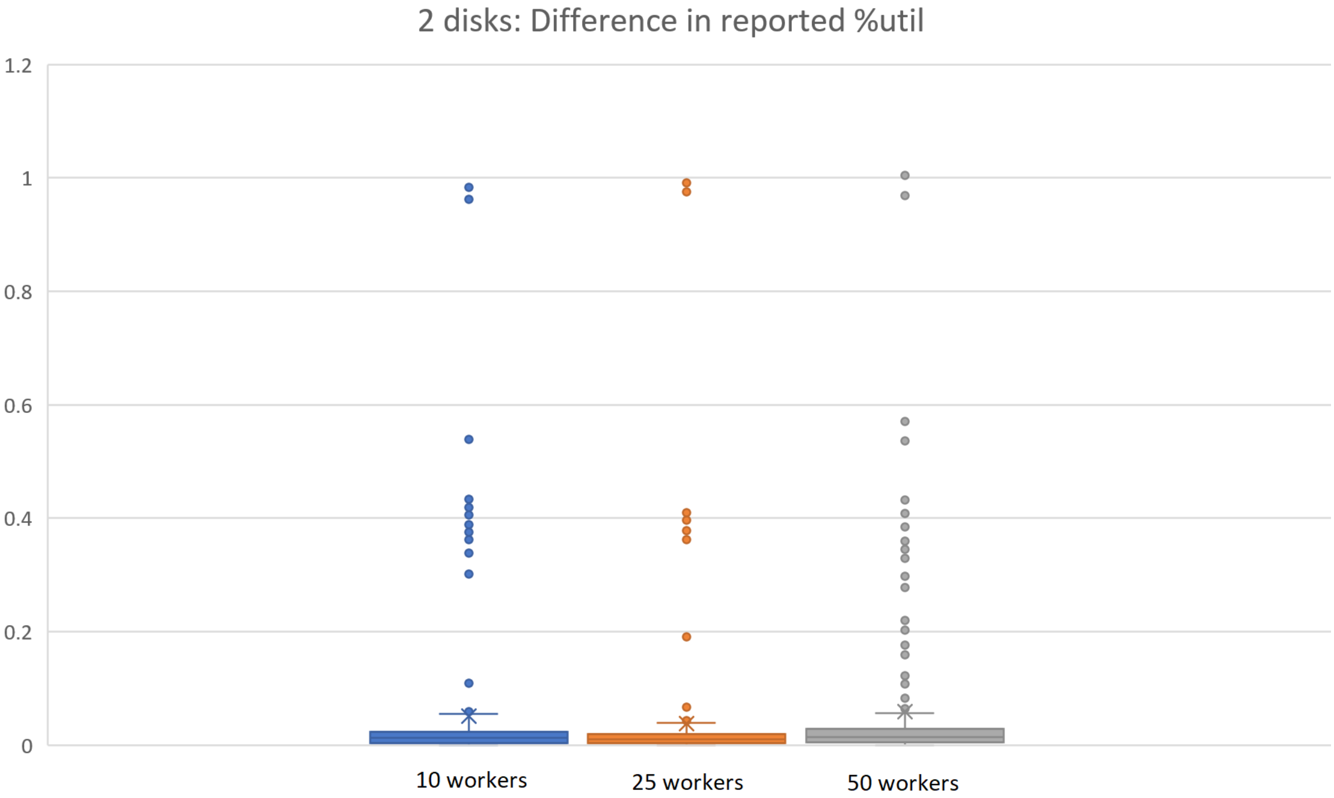
1 disk


1 disk: Latency of reading /proc/diskstats (ns)
3000000
2500000
2000000
1500000
1000000
500000
0
10 workers
25 workers
50 workers

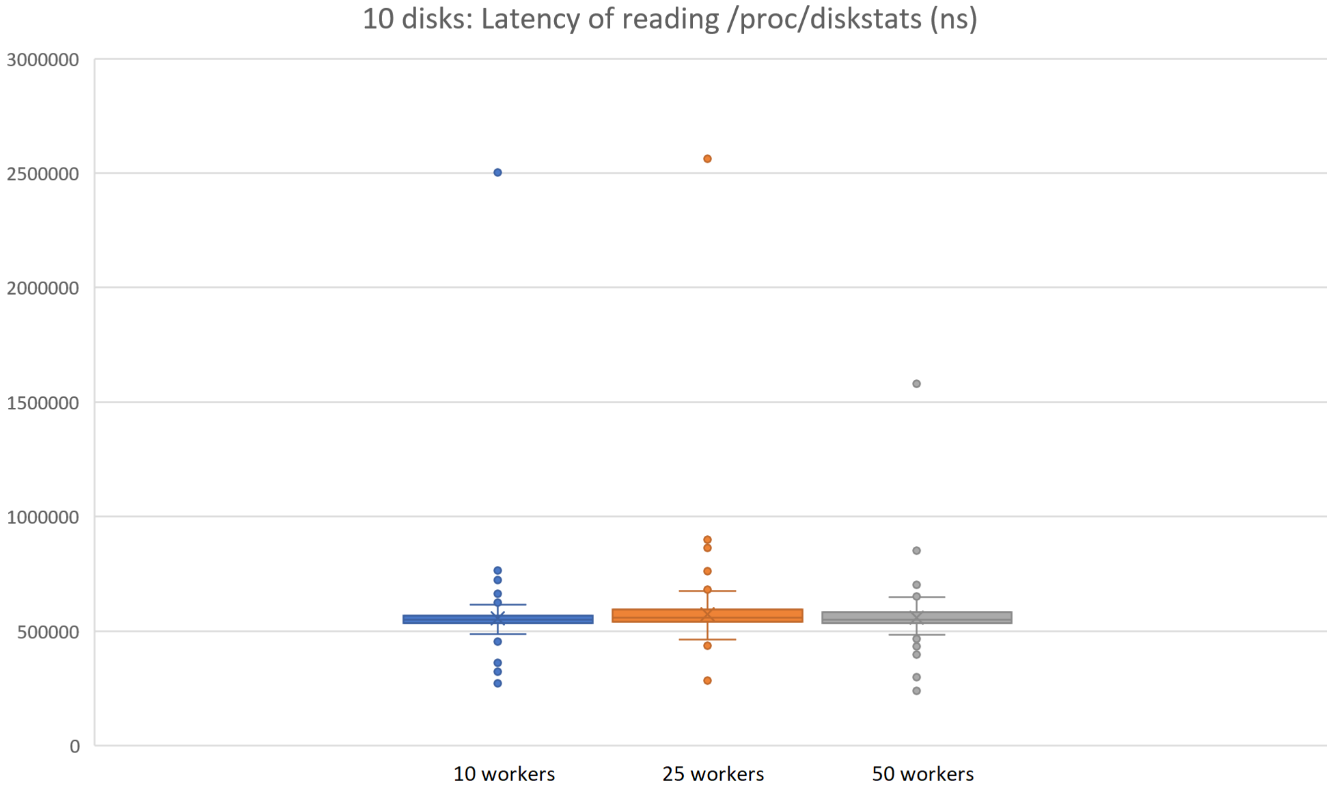

1 disk: Difference in reported % util
1.2
1
0.8
0.6
0.4
0.2
0
10 workers
25 workers
50 workers

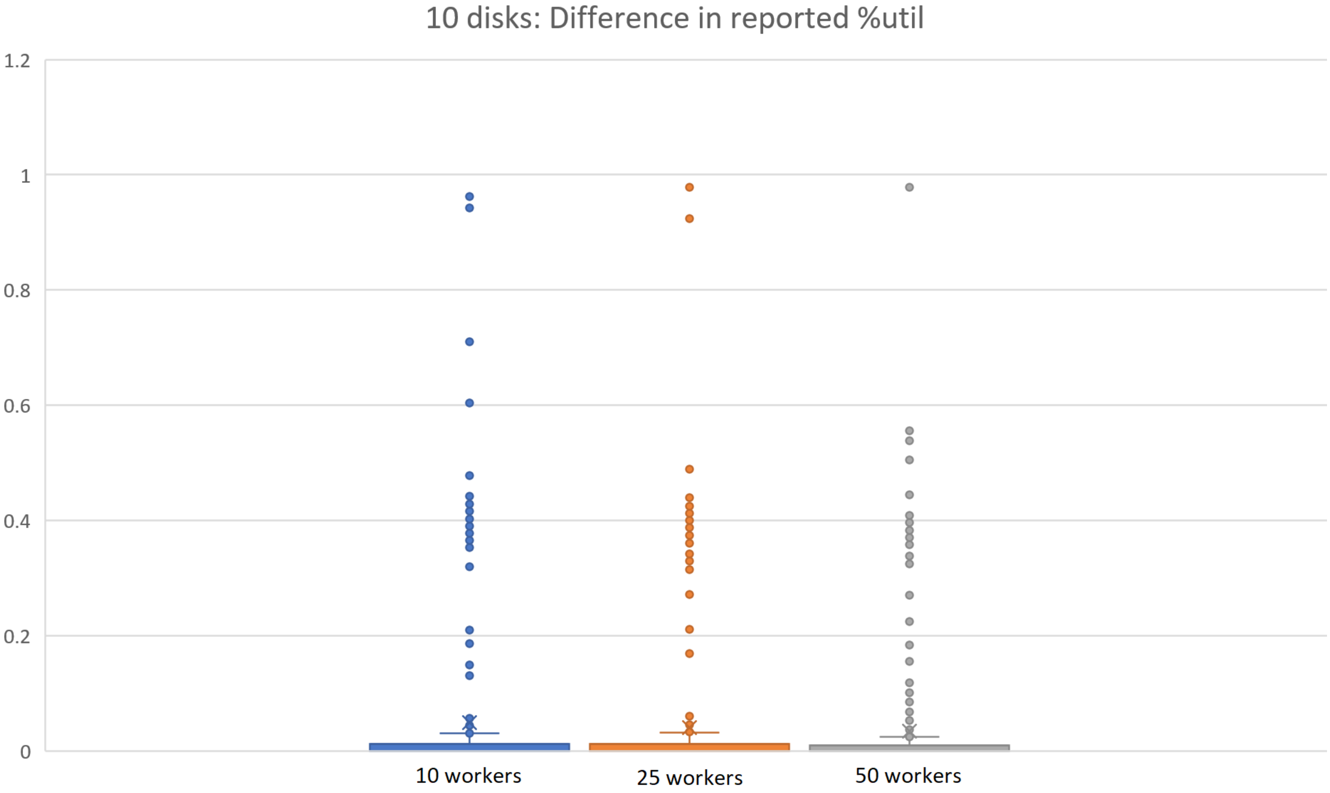
2 disks


2 disks: Latency of reading /proc/diskstats (ns)
2500000
2000000
1500000
1000000
500000
0
10 workers
25 workers
50 workers


2 disks: Difference in reported %util
1.2
1
0.8
0.6
0.4
0.2
0
10 workers
25 workers
50 workers

10 disks


10 disks: Latency of reading /proc/diskstats (ns)
3000000
2500000
2000000
1500000
1000000
500000
0
10 workers
25 workers
50 workers


10 disks: Difference in reported %util
1.2
1
0.8
0.6
0.4
0.2
0
10 workers
25 workers
50 workers

Conclusion: The average latency of **/proc/diskstats** is around 0.6 ms in all experiments except for the case with 1 disk and 50 workers, which has an average latency of around 0.4 ms. The difference in reported percentage disk utilization between **/proc/diskstats** and **iostats** is on average around 0.05%, except for the case with 1 disk and 50 workers, which has an average difference of 0.15%.