Linux System Tuning for Access Points

Greg Thain

Agenda

Goals of Tuning / This talk:

- Prevent Crashes
- Raise Throughput

Topics:

- Provisioning Memory
- Disk space and disk latency
- Sandbox transfers
- Cpus

Don't Panic!

Out of the box defaults work for many sites

Fewer than 10,000 jobs, probably ok

But many of you are running bigger sites…

Or are using VMs/container with knobs

The memory hotplug surprise

Kernel autosizes most knobs appropriately

fds, disk caches, process limits, etc.

…Except when vm memory hotplug enabled.

Use worker node for AP?

Works fine for small sites,

But if you have the budget, here's what to do…

Biggest problem with Access Points?

Users!

```
$ ssh head-node
Password:
Success. Logging you in...

Welcome to Ubuntu 18.04.6 LTS
```

Running jobs directly on the head node will not be tolerated and will lead to the **immediate suspension** of your account to protect the scheduler.

Users (generally) want to do right thing

"AP is a shared resource, treat it like one"

"Use some of the AP, but not too much" - ???

```
$ uptime
12:16:52 up 53 days,35 users,load average: 12.49, 10.98, 12.07
$ ps auxwwr
gthain 1033  0.0  1364252 488980 R 126:11 /home/gthain/.vscode…-
gthain 1033  0.0  2364252 184980 R 225:01 /home/gthain/.vscode…-
gthain 1033  0.0  2360152 388980 R  26:11 /home/gthain/.vscode…-
gthain 1033  0.0  3364252 288980 R 326:11 /home/gthain/.vscode…-
gthain 1033  0.0  1004252 188980 R 426:11 /home/gthain/.vscode…-
…
```

Solution: ulimit the cpu

/etc/security/limits.d/30-cpus

```
# Prevent rogue user process
# Too many resources for too lo

*          soft    cpu     118
*          hard    cpu     120
root       soft    cpu     unlimited
root       hard    cpu     unlimited
```

Units are in MINUTES!

Memory

1st rule of provisioning AP Memory:

Memory

1st rule of provisioning AP Memory:

DO NOT ENABLE SWAPPING / PAGING

Memory

Schedd holds all* job ads IN MEMORY

*including held jobs!

Roughly 1Gb / 10,000 ads

One condor_shadow process per running job

Roughly 1 Mb / condor_shadow

we work hard to limit this (spoiler: hard)

Schedd Forks for queries!

Want 3x headroom in case fork gets bad

So (rule of thumb…)

$$(3 * jobs_{total} * 1Gb) + (jobs_{running} * 1Mb)$$

+ Whatever user processes need

CPUs

condor_schedd is (*mostly)* single-threaded

what does this mean?

Adding more cpus doesn't help schedd

One slow call holds up the whole thing

Leads to *DaemonCoreDutyCycle*

DaemonCoreDutyCycle

condor_status –af
RecentDaemonCoreDutyCyle

< 0.95 → Probably OK

≥ 0.95 → Probably too busy

DC status from 2023-09-19 14:59:05 to 2023-09-19 15:03:05:
Duty Cycle: 88.09% Ops/second: 40.375

Runtime stats from 2023-09-19 14:59:05 to 2023-09-19 15:03:05:

InstRt	InstAvg	TotAvg	TotMax	RtPctAvg	InstRate	AvgRate	Item
93.115	0.07747	0.13	27.98	59.7981	5.01	0.49	Timer_start_job
78.429	0.09472	0.10	27.94	99.2199	3.45	0.62	Create_ProcessTot
39.962	0.00978	0.03	107.81	36.4341	17.02	5.01	fsync
22.536	0.33142	0.22	63.14	150.6389	0.28	0.28	SCWalkJobQ
17.386	0.39513	0.26	5.37	152.7251	0.18	0.17	SCBuildPrioRec
16.506	0.37515	0.25	5.36	152.2462	0.18	0.17	SCWalkJobQ_get_job_prio
7.628	0.16582	0.08	436.30	211.2148	0.19	0.41	Timer_timeout
6.410	0.11245	0.18	21.41	62.6880	0.24	0.17	Timer_StartJobHandler
5.979	0.00000	0.00	0.15	101.7150	12196.33	9285.09	SCGetAutoCluster
4.558	0.00000	0.00	0.07	111.3440	12177.73	9265.80	SCGetAutoCluster_hit
4.352	0.43525	0.31	2.35	140.5082	0.04	0.05	SCWalkJobQ_count_a_job
3.994	0.01035	0.01	19.10	119.8948	1.61	0.62	Signal_HandleDC_SERVICEWAITPIDS
2.095	0.00858	0.01	0.28	160.8964	1.02	0.22	Timer_checkContactQueue
1.420	0.00032	0.00	0.15	108.3520	18.60	19.29	SCGetAutoCluster_signature
1.100	0.09167	0.02	1.17	473.9021	0.05	0.08	Timer_TokenRequesttryTokenRequests
1.001	0.10009	0.06	0.68	171.1175	0.04	0.05	SCWalkJobQ_check_for_spool_zombies
0.993	0.00229	0.00	0.65	102.5686	1.80	0.66	Signal_HandleDC_SIGCHLD
0.699	0.00065	0.00	0.25	312.8221	4.48	1.91	Timer_CheckTransferQueue
0.677	0.16925	0.14	63.14	116.8083	0.02	0.02	Timer_PeriodicExpr
0.676	0.16912	0.14	63.14	116.8294	0.02	0.02	SCWalkJobQ_PeriodicExprEval
0.621	0.00053	0.00	3.46	89.0456	4.84	1.28	Command_TRANSFER_QUEUE_REQUEST
0.619	0.00010	0.00	25.55	108.1283	26.82	14.36	Command_QMGMT_WRITE_CMD
0.577	0.01310	0.01	2.43	188.7987	0.18	0.17	SCBuildPrioRec_sort
0.558	0.00237	0.00	4.94	84.6769	0.98	0.53	Timer_SelfDrainingQueuetimerHandlerjob_is_finished_

Two blocking calls to know

1. Slow DNS queries

2. Slow fsync

Slow DNS queries in SchedLog

```
09/12/23 10:14:02 (D_ALWAYS)
WARNING: Saw slow DNS query,
which may impact entire system:
getaddrinfo(cm.chtc.wisc.edu)
took 24.445889 seconds.
```

Slow fsync

09/19/23 00:37:06 (pid:2903754) (D_ALWAYS:2) Transaction::Commit(): fdatasync() took 11 seconds to run

Schedd not only user of CPU

Xfer'ing shadow uses 5-10% of a cpu*

Very little otherwise

User work also uses cpus

So provision cpus on APs accordingly

Disk: Two aspects

1. Space

2. Latency (not throughput)

SPOOL

All spooled files (including checkpoints) go in

condor_config_val SPOOL

Bad things will happen if this fills up.

Ideally on separate partition

Can configure a per-user SPOOL with

…

job_queue.log

(sorry about the name log)

The schedd's transaction log

FSYNC'd very frequently

can slow down whole schedd

Fsync and ext3

Fsync (actually fdatasync) call on ext3

can be very slow

Please use JOB_QUEUE_LOG to put

job_queue.log on own partition

AND if one nvme/ssd drive, put it there.

May be biggest thing you can do for the AP

Networking

Assuming condor file transfer..

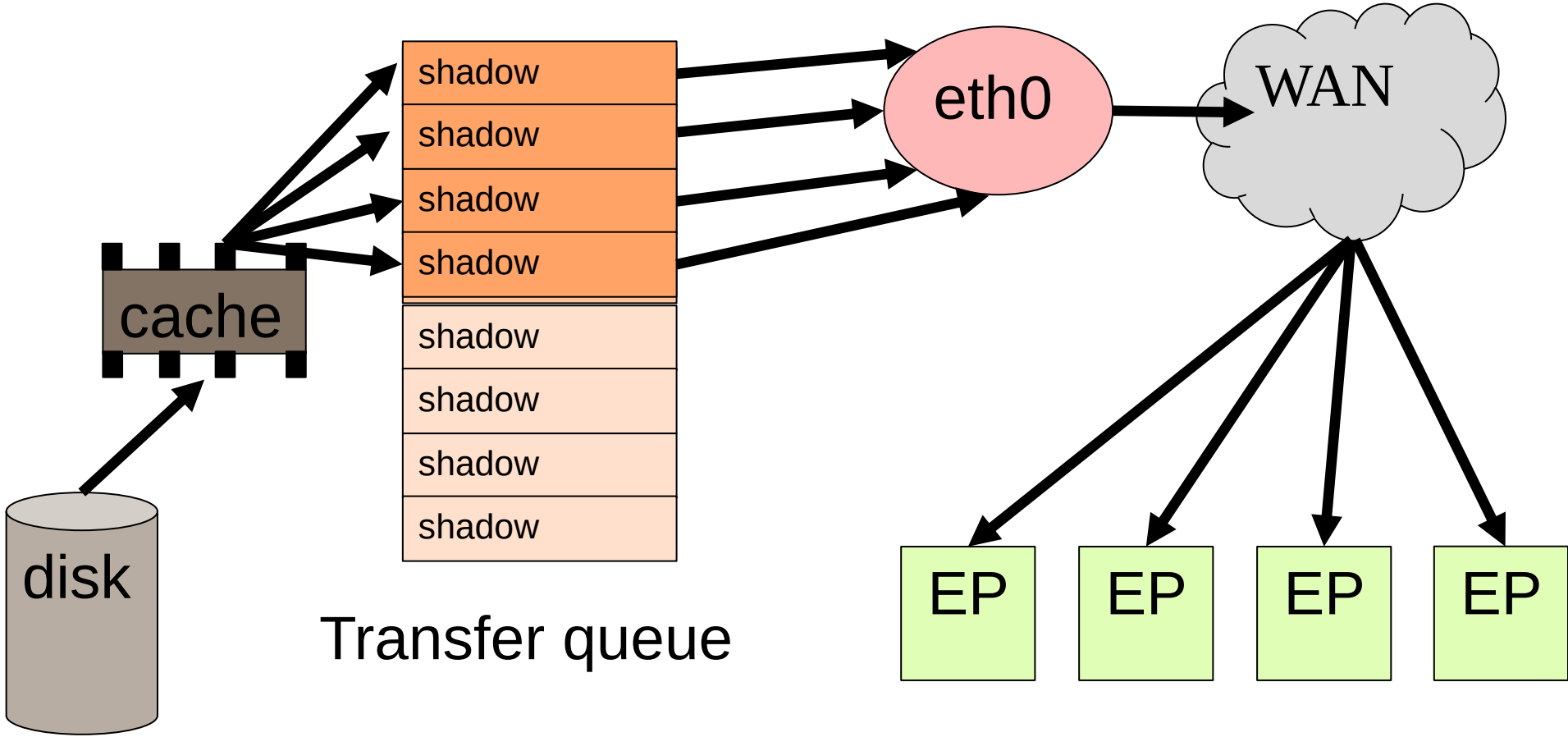
More is better – 10Gb, 100+Gb

Install fq_codel if not already on

Think about ratio of AP to EP and

# of concurrent transfers

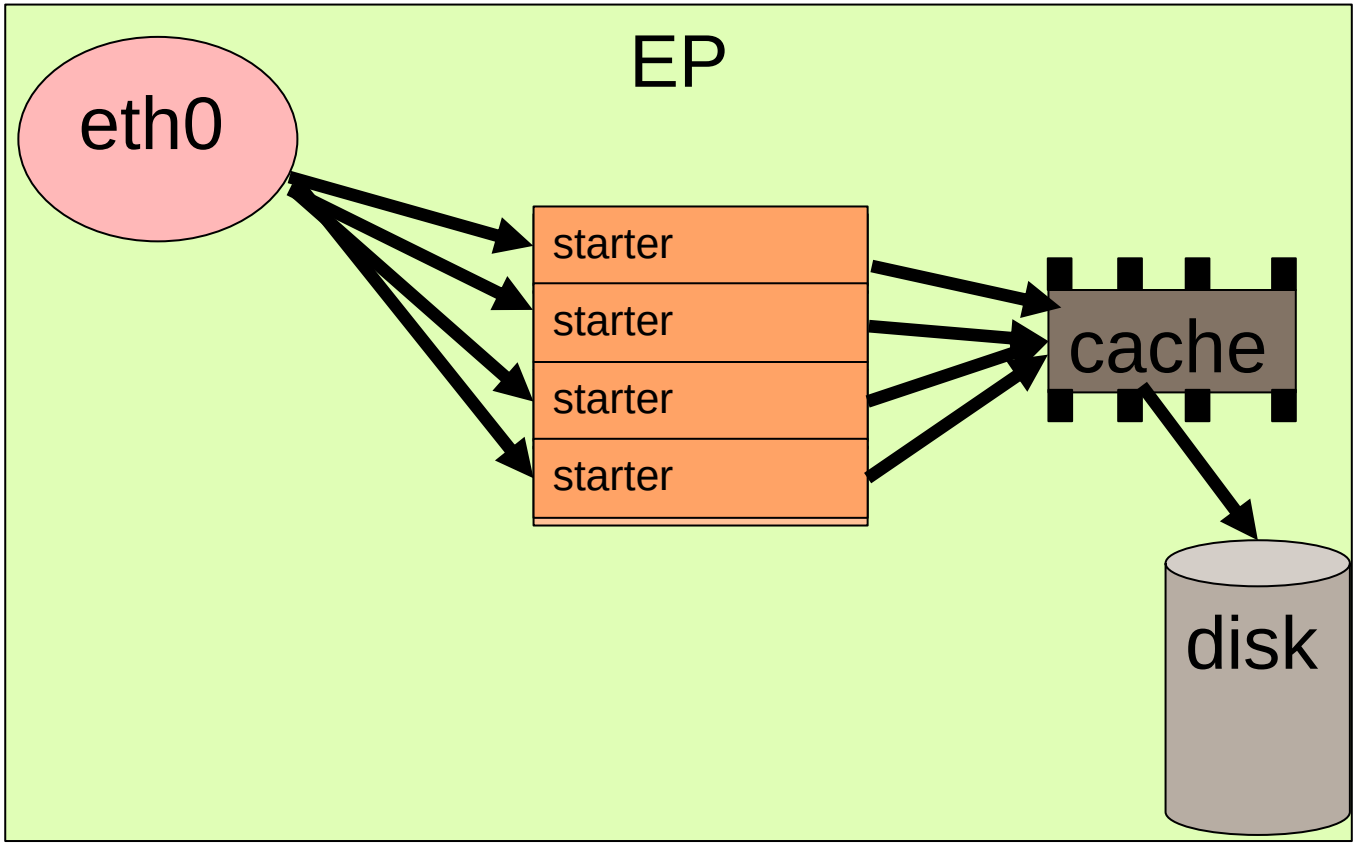
File Transfer Model



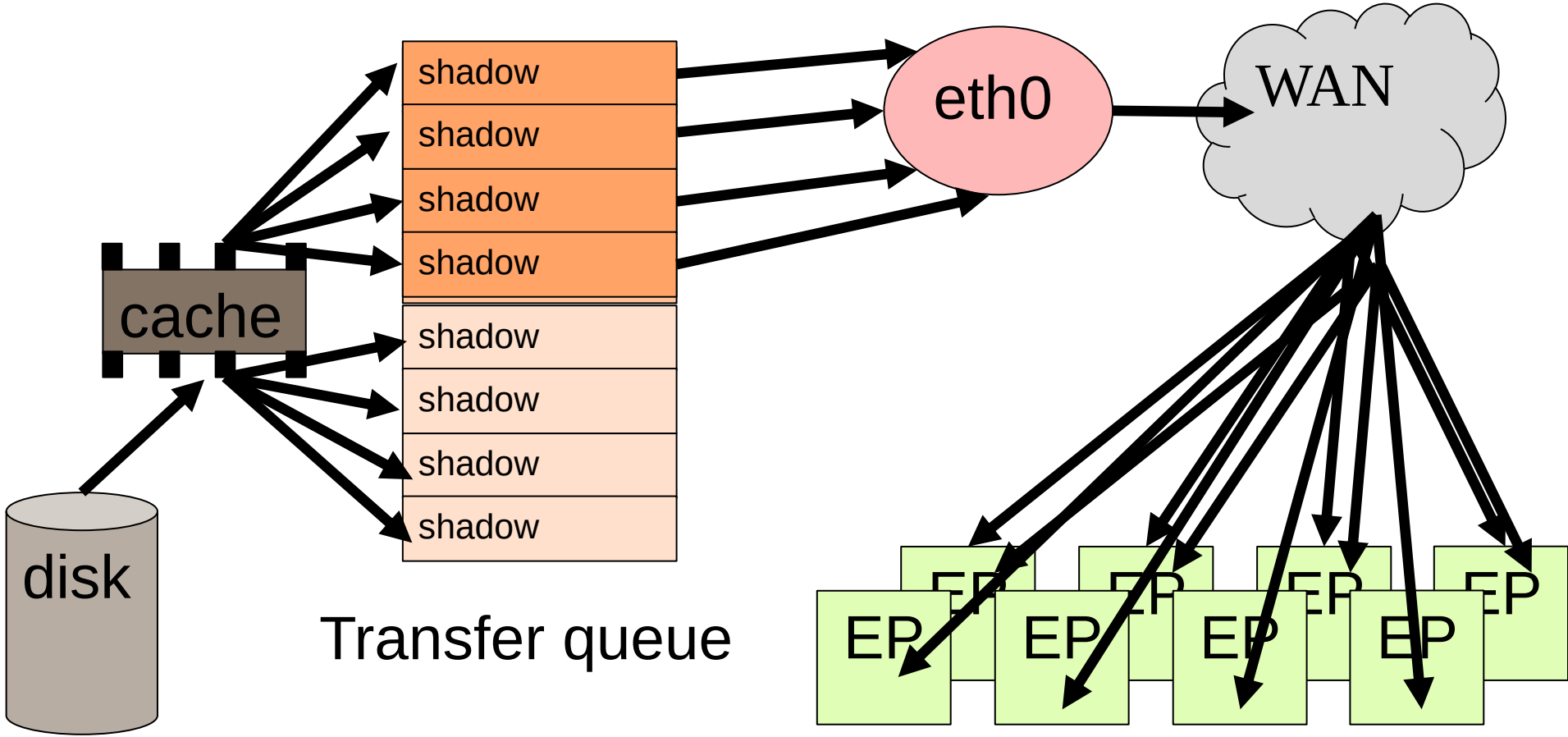
File Transfer Model

EP

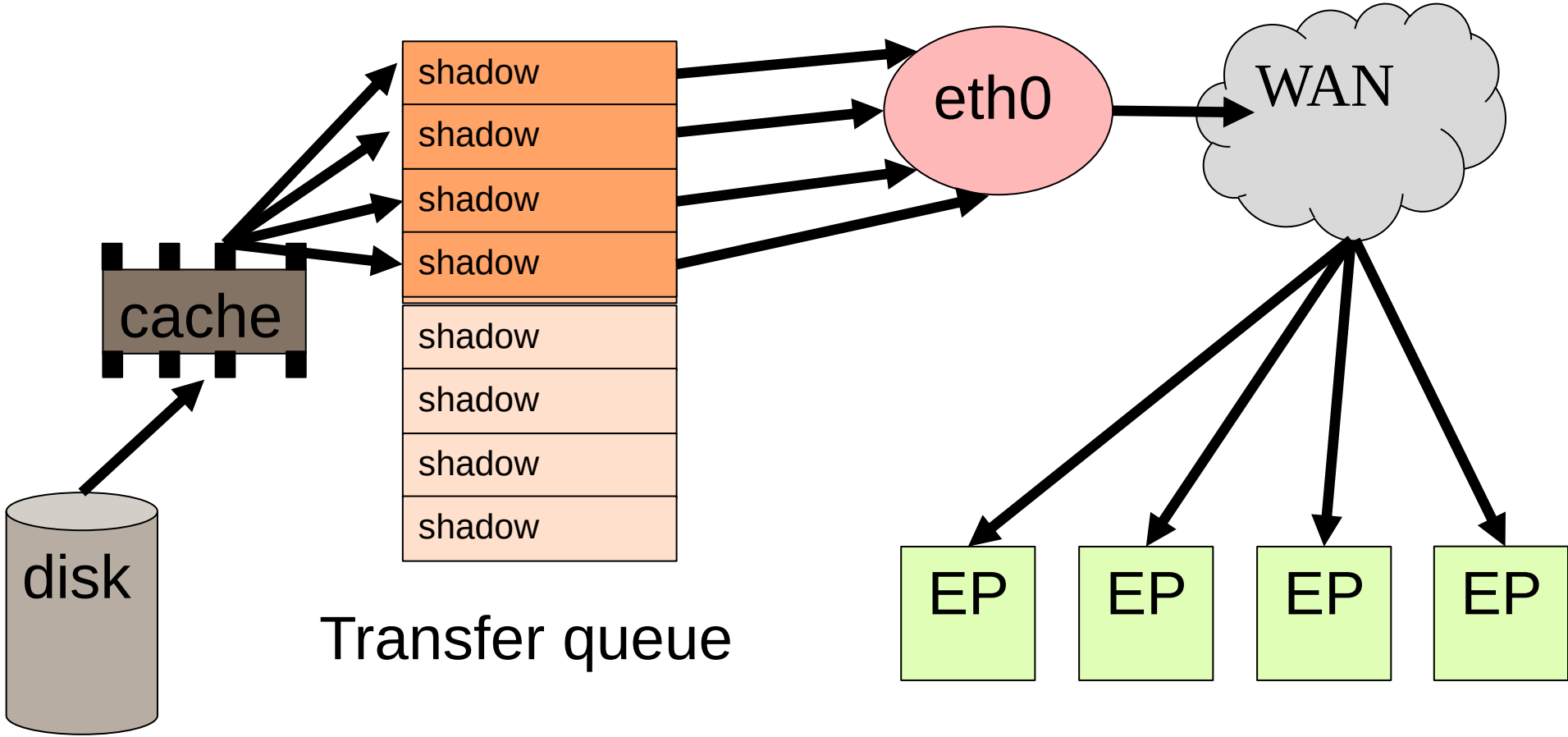
File Transfer (EP side)



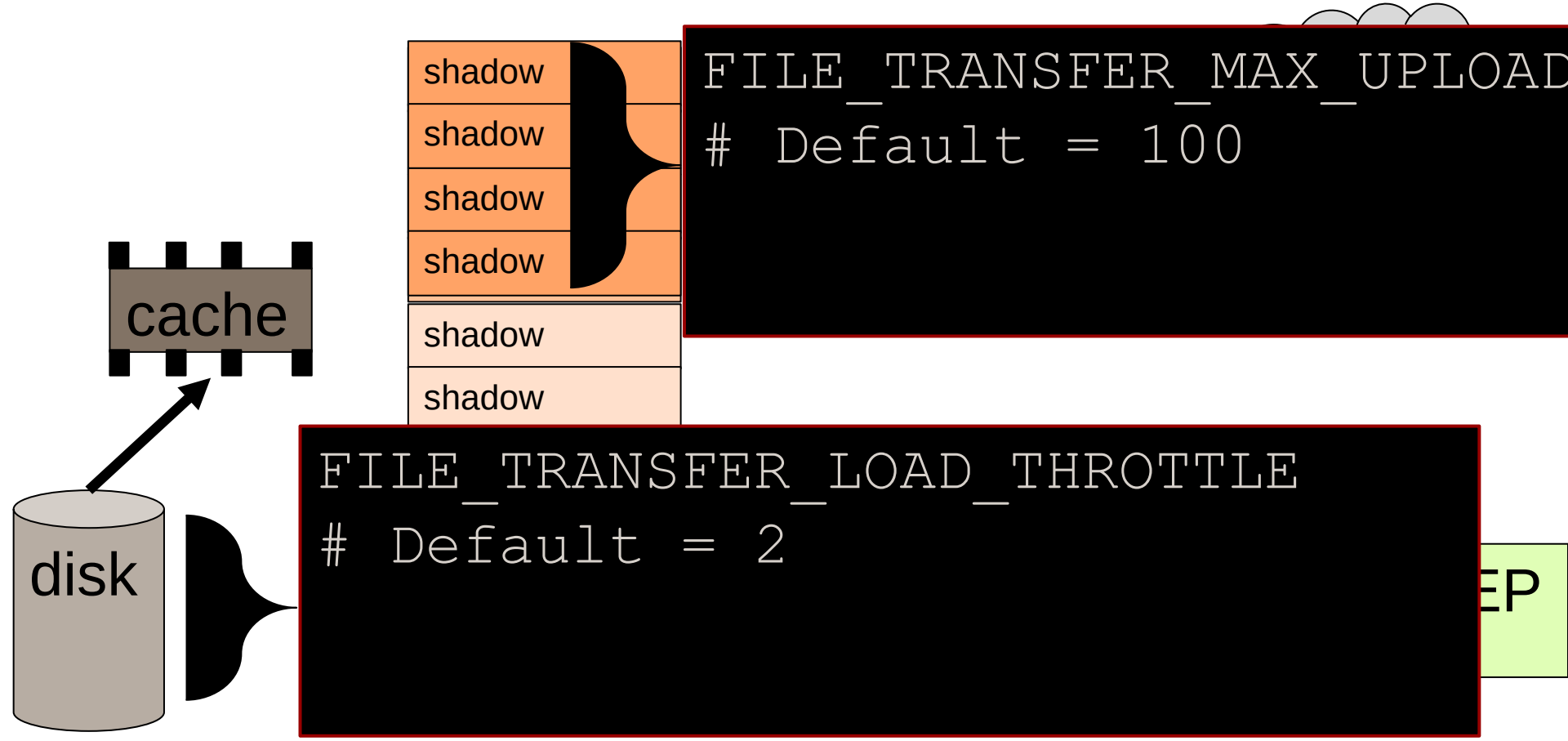
File Transfer – WHY?



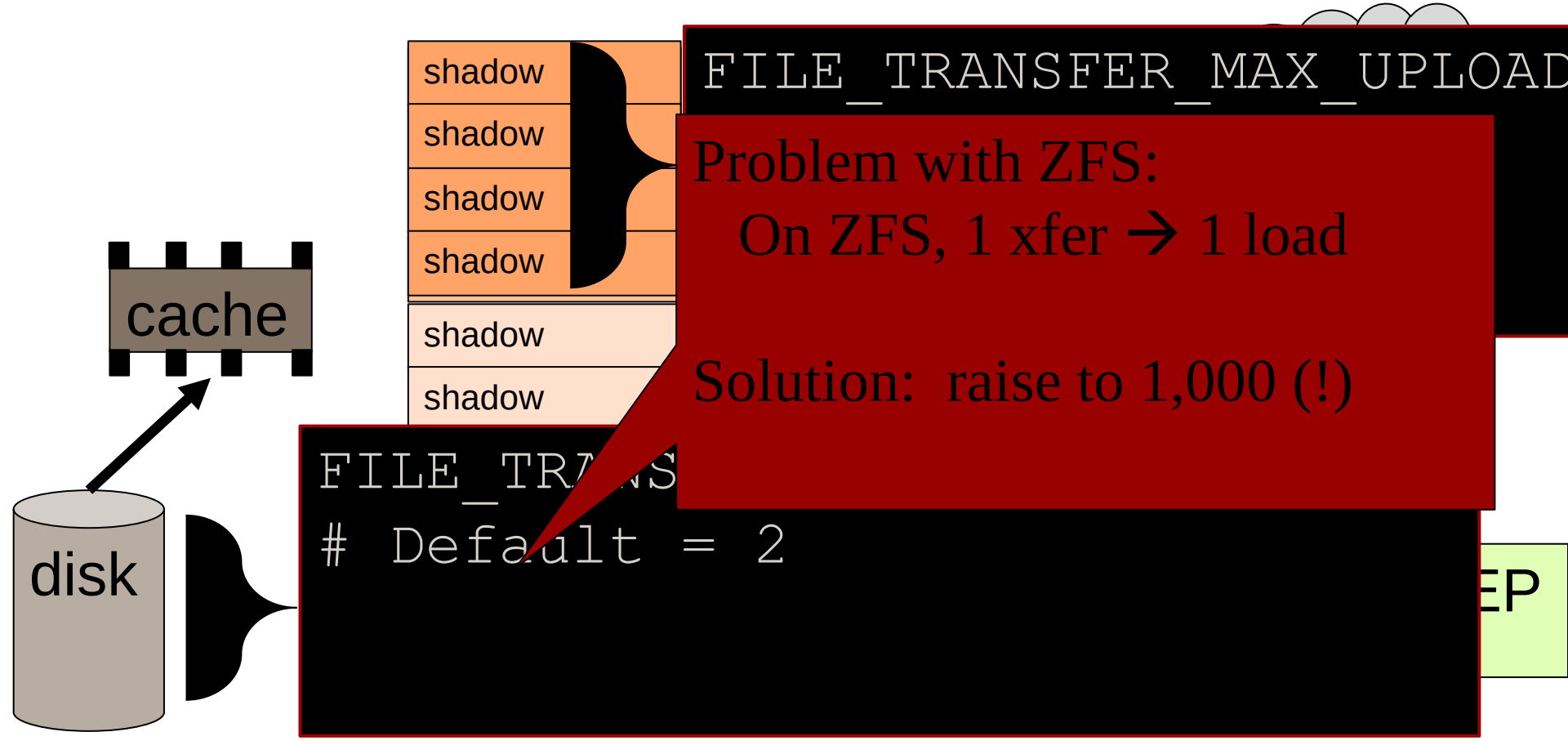
File Transfer Speeds



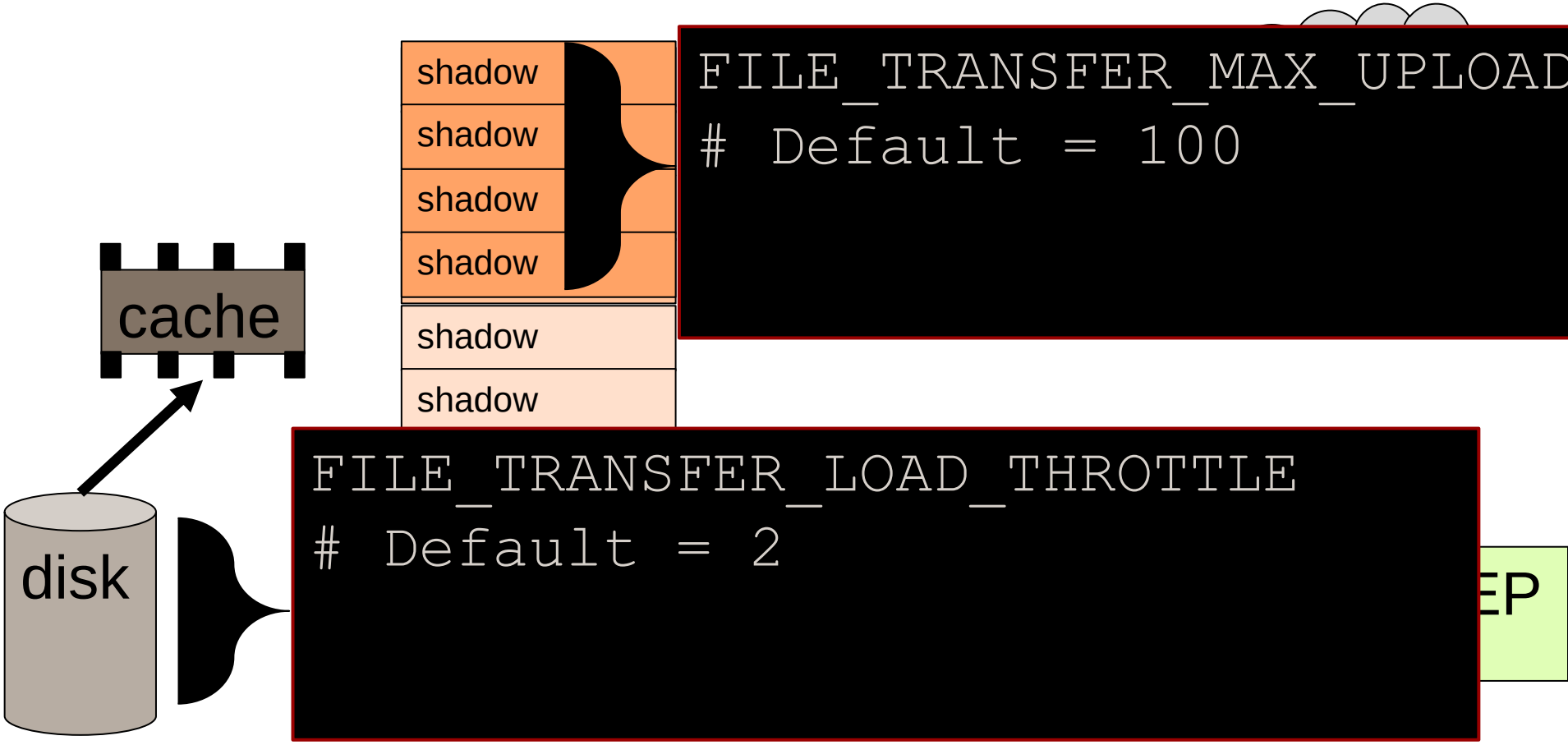
HTCondor's throttles



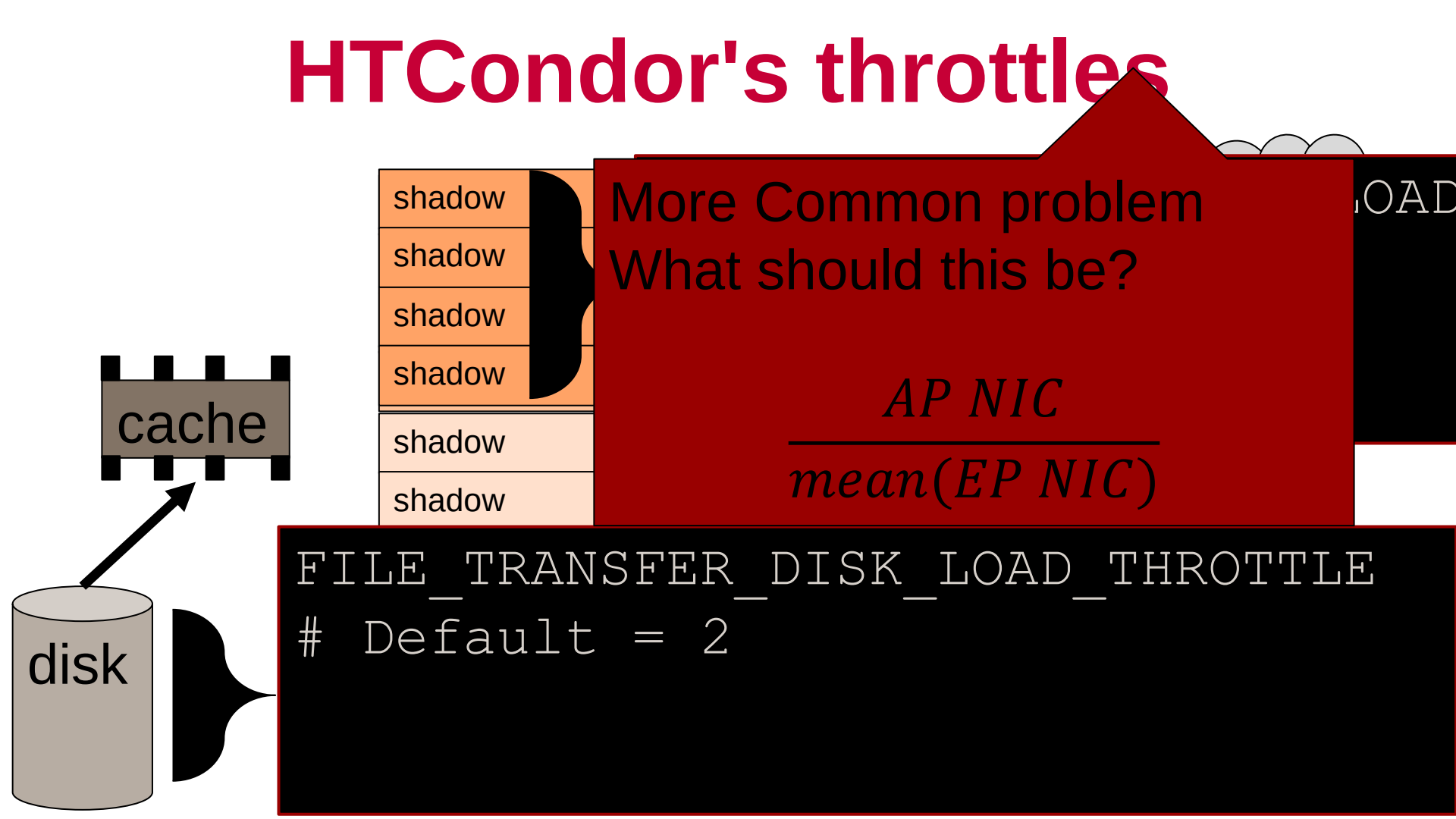
HTCondor's throttles



HTCondor's throttles



HTCondor's throttles



Things to look at

```
$ condor_status -schedd -l name_of_your_schedd | grep Transfer
FileTransferMBWaitingToUpload = 0.0
FileTransferUploadBytes = 181516308696053.0
FileTransferUploadBytesPerSecond_1d = 64747373.05902614
FileTransferUploadBytesPerSecond_1h = 215636610.644708
FileTransferUploadBytesPerSecond_1m = 56930145.62086731
FileTransferUploadBytesPerSecond_5m = 120896405.913133
TransferQueueMaxUploading = 100
TransferQueueNumUploading = 10
TransferQueueNumUploadingPeak = 100
TransferQueueNumWaitingToUpload = 0
TransferQueueNumWaitingToUploadPeak = 5582
TransferQueueUploadWaitTime = 0
TransferQueueUploadWaitTimePeak = 11482
```

Thank you!

Remember:

- Don't panic about modest APs
- Put job_queue.log on own disk (nvme?)
- Keep monitoring your AP
  - Especially DaemonCoreDutyCycle

Questions?