POSH: A Data-Aware Shell

Deepti Raghavan, Sadjad Fouladi, Philip Levis, Matei Zaharia

Stanford University

USENIX ATC 2020

Shell Data Processing

```
deeptir@dash ~/stanford/cs448b/project cat ssh_data_viz.csv | grep '135.87.0.0'   ✔ 10111 11:55:06
135.87.0.17,10455,135.87.0.0/24,135.87.0.0,37.751,-97.822,2,2,2,2,2,2,2,2,135.87.0.0,135.0.0.0
135.87.0.204,10455,135.87.0.0/24,135.87.0.0,37.751,-97.822,2,2,2,2,2,2,2,2,135.87.0.0,135.0.0.0
135.87.0.237,10455,135.87.0.0/24,135.87.0.0,37.751,-97.822,2,2,2,2,2,2,2,2,135.87.0.0,135.0.0.0
135.87.0.28,10455,135.87.0.0/24,135.87.0.0,37.751,-97.822,2,2,2,2,2,2,2,2,135.87.0.0,135.0.0.0
135.87.0.221,10455,135.87.0.0/24,135.87.0.0,37.751,-97.822,2,2,2,2,2,2,2,2,135.87.0.0,135.0.0.0
135.87.0.29,10455,135.87.0.0/24,135.87.0.0,37.751,-97.822,2,2,2,2,2,2,2,0,135.87.0.0,135.0.0.0
135.87.0.29,10455,135.87.0.0/24,135.87.0.0,37.751,-97.822,2,2,2,2,2,2,2,0,135.87.0.0,135.0.0.0
135.87.0.205,10455,135.87.0.0/24,135.87.0.0,37.751,-97.822,2,2,2,2,2,2,0,2,135.87.0.0,135.0.0.0
135.87.0.236,10455,135.87.0.0/24,135.87.0.0,37.751,-97.822,2,2,2,2,2,2,2,2,135.87.0.0,135.0.0.0
135.87.0.25,10455,135.87.0.0/24,135.87.0.0,37.751,-97.822,2,2,2,2,2,2,2,2,135.87.0.0,135.0.0.0
135.87.0.36,10455,135.87.0.0/24,135.87.0.0,37.751,-97.822,2,2,2,2,2,2,2,2,135.87.0.0,135.0.0.0
135.87.0.37,10455,135.87.0.0/24,135.87.0.0,37.751,-97.822,2,2,2,2,2,2,2,2,135.87.0.0,135.0.0.0
135.87.0.220,10455,135.87.0.0/24,135.87.0.0,37.751,-97.822,2,2,2,2,2,2,2,2,135.87.0.0,135.0.0.0
135.87.0.45,10455,135.87.0.0/24,135.87.0.0,37.751,-97.822,2,2,2,2,2,2,2,2,135.87.0.0,135.0.0.0
135.87.0.20,10455,135.87.0.0/24,135.87.0.0,37.751,-97.822,2,2,2,2,2,2,2,2,135.87.0.0,135.0.0.0
135.87.0.21,10455,135.87.0.0/24,135.87.0.0,37.751,-97.822,2,2,2,2,2,2,2,2,135.87.0.0,135.0.0.0
135.87.0.44,10455,135.87.0.0/24,135.87.0.0,37.751,-97.822,2,2,2,2,2,2,2,2,135.87.0.0,135.0.0.0
deeptir@dash ~/stanford/cs448b/project                                          ✔ 10112 11:55:52
```

```
git log                                                                    ⌥⌘3
commit 53bb6523df808fc848fd3b66acc2df69be6a9ed1 (HEAD -> tests, origin/tests)
Author: deeptir <deeptir@cs.stanford.edu>
Date:   Thu Mar 5 12:20:05 2020 -0800

    Add unit tests

commit 33fcb4b4d4e62ae4c51487f577812cc0f7f016bd
Author: deeptir <deeptir@cs.stanford.edu>
Date:   Thu Mar 5 12:19:55 2020 -0800

    Add utilities for equality and recursive walking of the tree; add Add rul
e

commit 4edaad4bde63195cdbde6964907692f49b528b3b (origin/master, origin/HEAD,
:
```

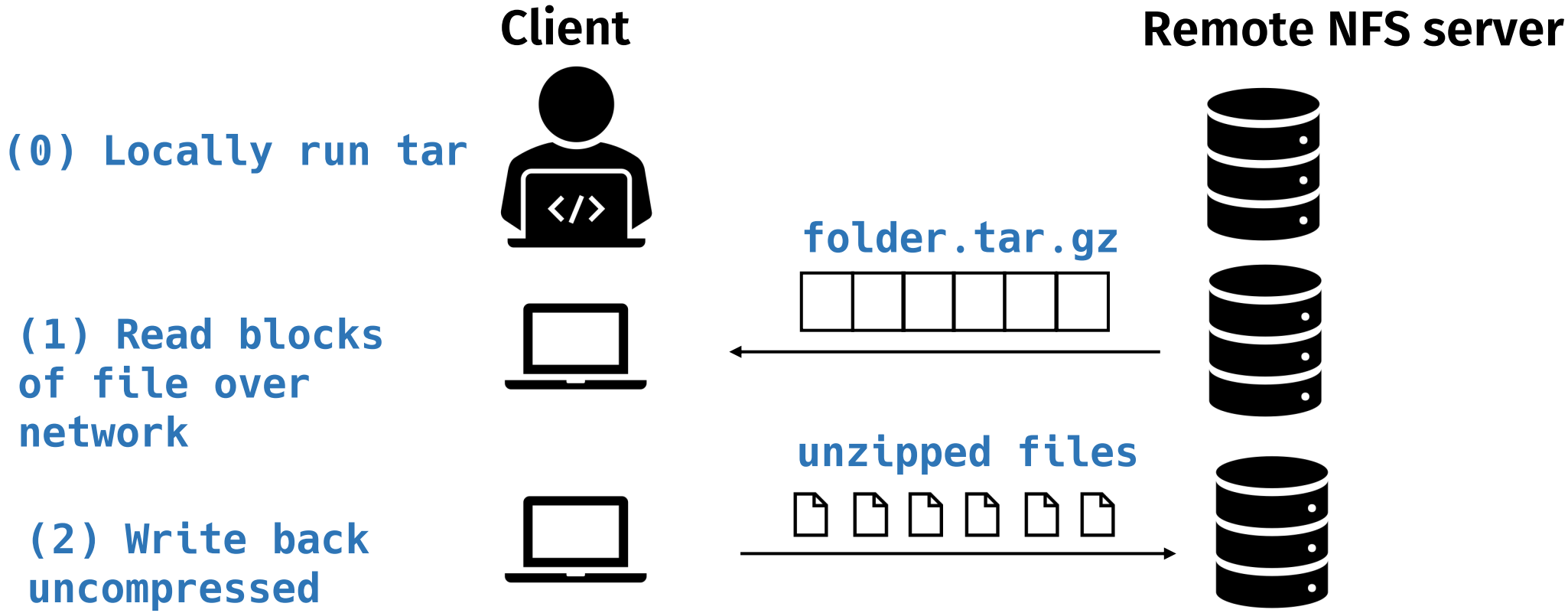

```
convert -size 100x100
```

Running Shell Commands over Remote Data

- Remote filesystems enable us to *preserve our local environment*
- Slowdown for using NFS, for a client and server in the *same GCP region,* operating on a *.5 Gb folder:*
  - Tar extract: **35x** slower
  - Git clone: **6x** slower
  - Copy recursive: **100x** slower

Bash Commands over NFS

```
tar -xzf folder.tar.gz
```



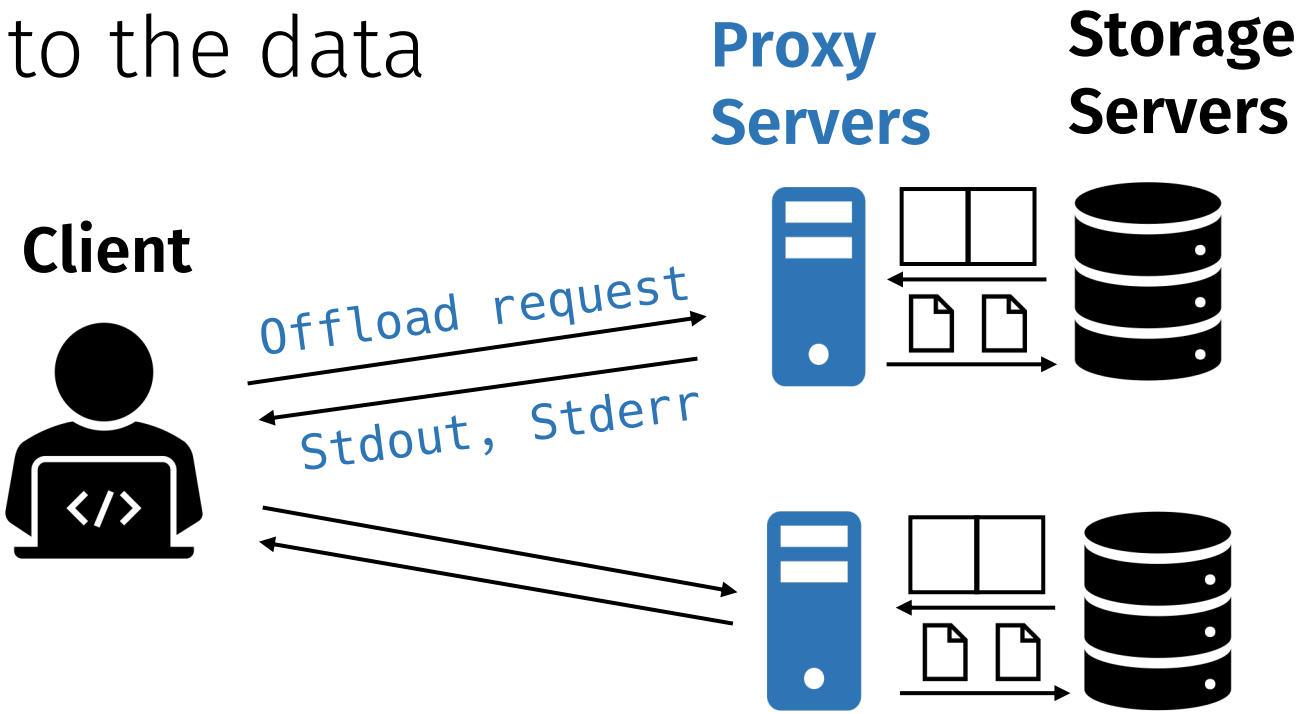
Near Data Computing

- Prior systems (Spark, Map Reduce, Active Disks, stored procedures) push computation closer to the data
  - Users must conform to a specific API
  - Overhead to accelerate simple shell pipelines that use remote filesystems

```
cat A B C D | grep “foo” | tee local.txt
```

Can we bring data locality to the shell?

Posh, the **Process Offload Shell:** Automatically offloads portions of *unmodified* shell pipelines to *proxy servers* closer to the data



bash$

posh>>>

Challenge & Contributions

- **Key Challenge:**
  - How does Posh know how to split computation between local execution and proxy servers?
- **Contributions:**
  1. An **annotation language** that summarizes the command-line semantics of each command and file dependencies
  2. A **runtime** that allows Posh to execute and parallelize shell commands across proxy servers
  3. A **scheduling algorithm** that lets Posh decided how to optimally offload and schedule the commands across proxy servers to reduce execution time

Agenda

- Introduction
- Annotations
- System Design
- Scheduling Algorithm
- Evaluation

What would an annotation language show?

Given an arbitrary command line, what information does Posh need to decide what to offload?

```
cat A B C D | grep “foo” | tee output.txt
```

1. The file dependencies of each command
2. How much data flows between commands
3. Which commands can be parallelized safely

Determining File Dependencies of a Command

Consider these command lines:

1. `cat A B C D | grep “foo”`
2. `tar -czf file.tar.gz input_folder`
3. `tar -xzf file.tar.gz <implicitly .>`
4. `git status <implicitly .>`

Annotation Interface

```
                    tar -czf file.tar.gz input_folder
tar:
  FLAGS:                  list of flags
    - short:c
    - short:z
  OPTPARAMS:              List of parameters preceded by flags
    - short:f, long:file, type:output_file, size:1
  PARAMS:                 List of parameters not preceded by flags
    - type:input_file, size:list
```

Annotation Interface

```
                    tar -czf file.tar.gz input_folder
tar:
  FLAGS:
    - short:c
    - short:z
  OPTPARAMS:
    - short:f, long:file, type:output_file, size:1
  PARAMS:
    - type:input_file, size:list
```

Annotation Interface

```
tar -czf file.tar.gz input_folder
tar:
  FLAGS:
    - short:c
    - short:z
  OPTPARAMS:
    - short:f, long:file, type:output_file, size:1
  PARAMS:
    - type:input_file, size:list
```

```
tar -xzf file.tar.gz
tar [needs_current_dir]:
  FLAGS:
    - short:x
    - short:z
  OPTPARAMS:
    - short:f, long:file, type:input_file, size:1
```

extra information about the entire command

different type assignment

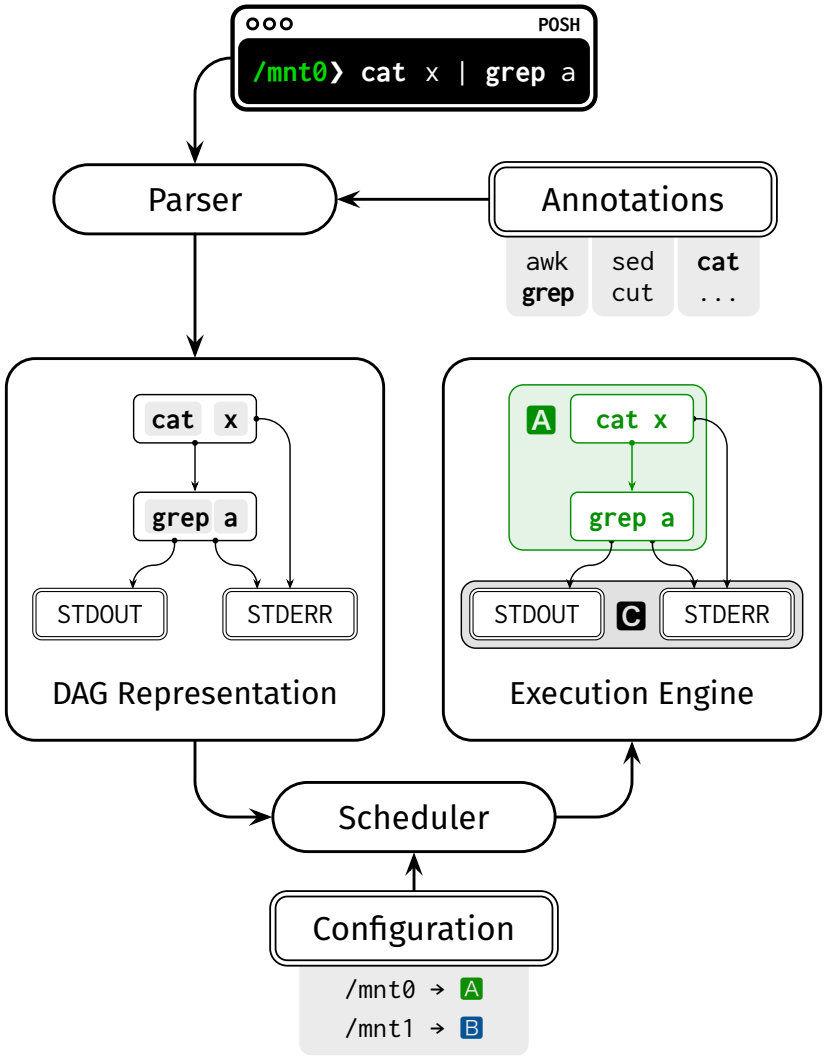
Shell Annotations

- Argument specific information:
  - **type** assignment: `input_file,output_file`
  - Whether command can be **parallelized** across this argument
- Overall command semantics:
  - **Parallelization** semantics
  - **Filtering** semantics
  - Implicitly needs current directory

Agenda

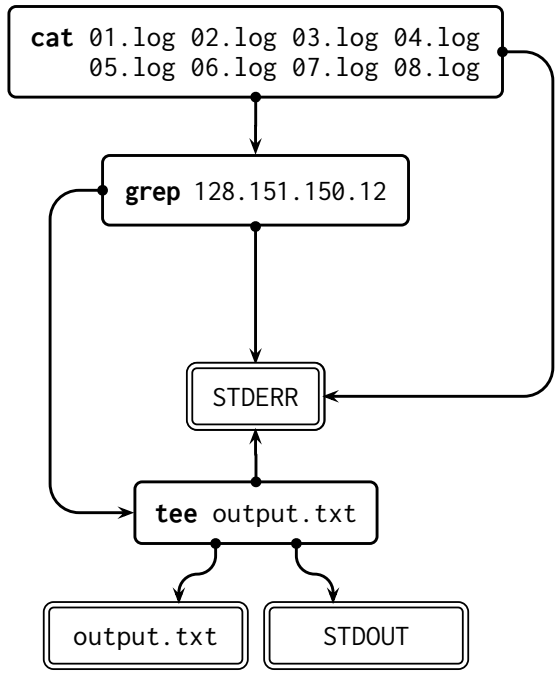
- Introduction
- Annotations
- System Design
- Scheduling Algorithm
- Evaluation

System Overview



Scheduling

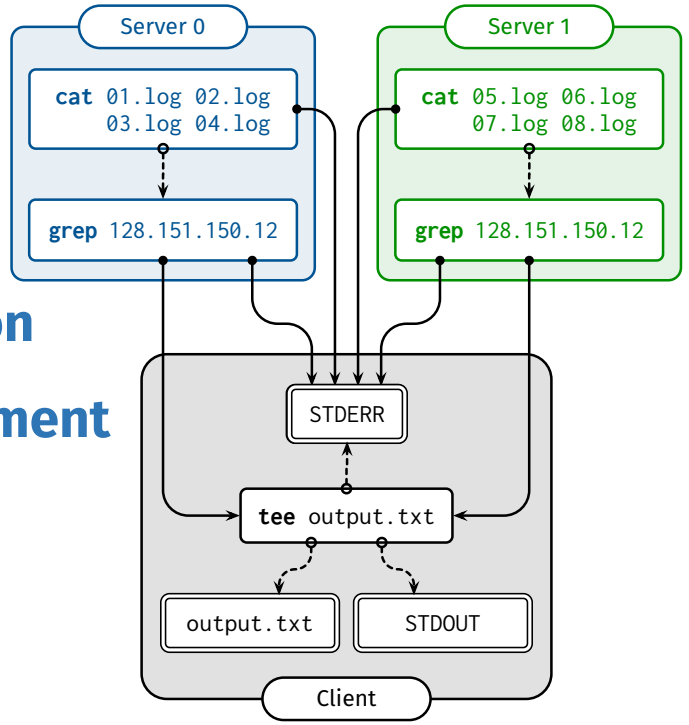
```
$ cat mount0/*.log mount1/*.log
  | grep 128.151.150.12
  | tee output.txt
```



(1) Automatic Parallelization

(2) Greedy Location Assignment

```
$ cat mount0/*.log mount1/*.log
  | grep 128.151.150.12
  | tee output.txt
```



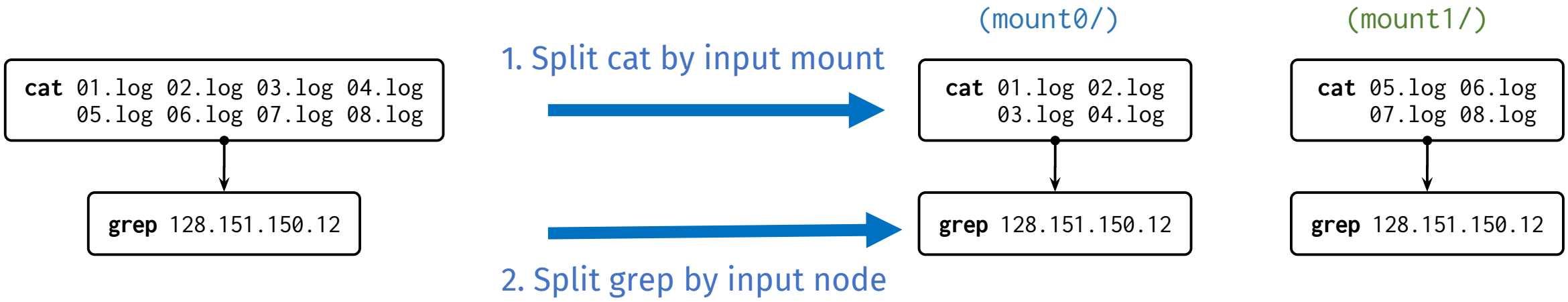
Automatic Parallelization

```
cat A B C D | grep "foo"
cat:
  PARAMS:
    - type:input_file,size:list,splittable

grep [splittable_across_input, filters_input]:
  PARAMS:
    - type:string,size:1
    - type:input_file,size:list
```

Automatic Parallelization

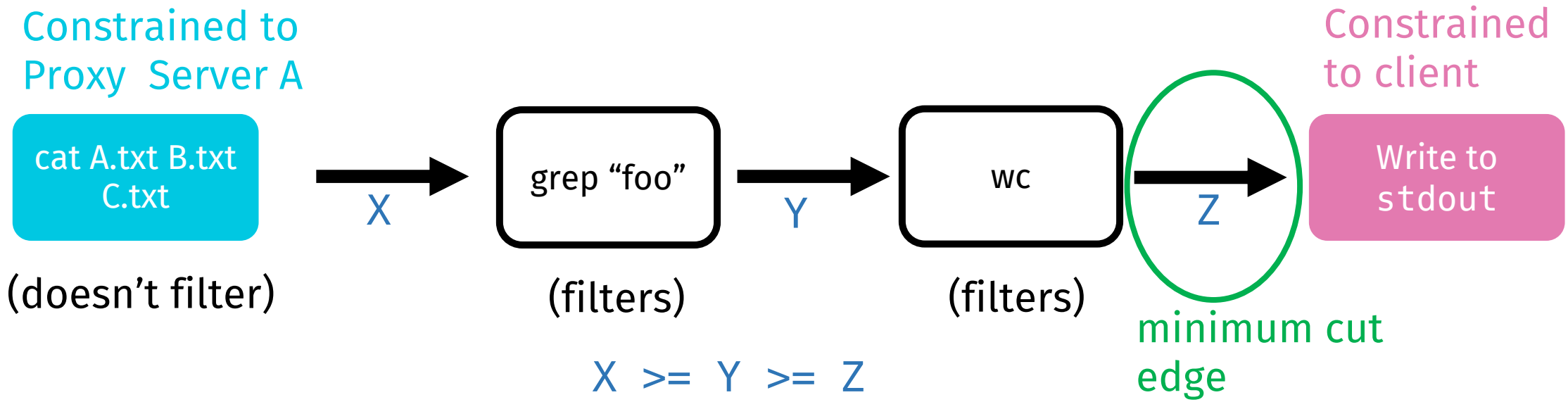
```
$ cat mount0/*.log mount1/*.log
  | grep 128.151.150.12
  | tee output.txt
```



Overview of Greedy Location Assignment

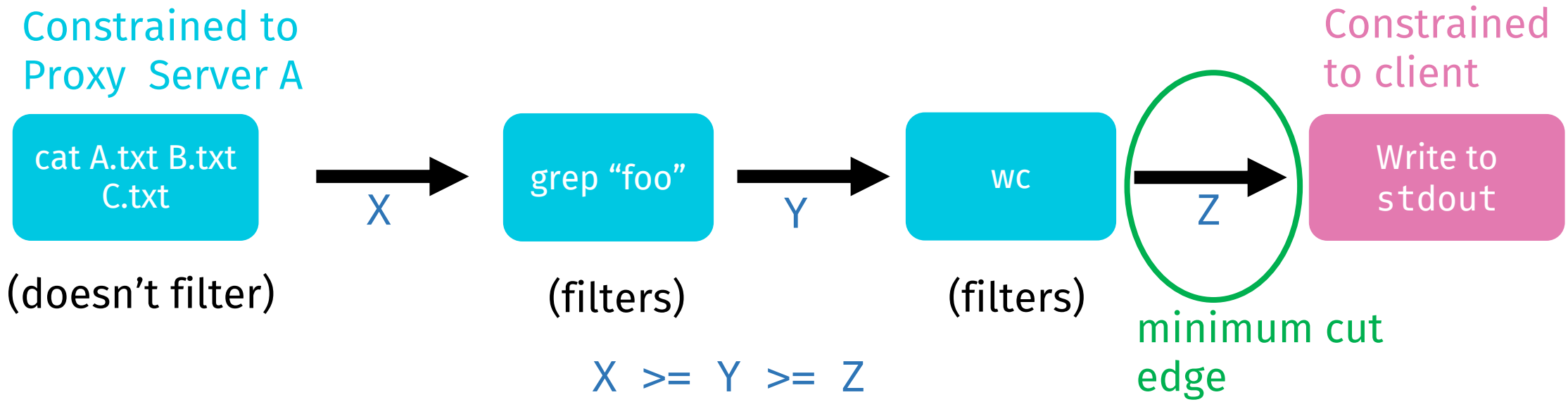
1. Assign nodes with constraints:
   - Read and write nodes
   - Nodes whose file dependencies all live on the same mount
   - Nodes who access files from different mounts **(assign to client)**
2. Consider each source -> sink path in the DAG
   - Uses heuristics to assign each unassigned node along a path
3. Resolve conflicting assignments

Greedy Location Assignment




```
grep [splittable_across_input, filters_input]:
```

Greedy Location Assignment




```
grep [splittable_across_input, filters_input]:
```

Implementation & Evaluation

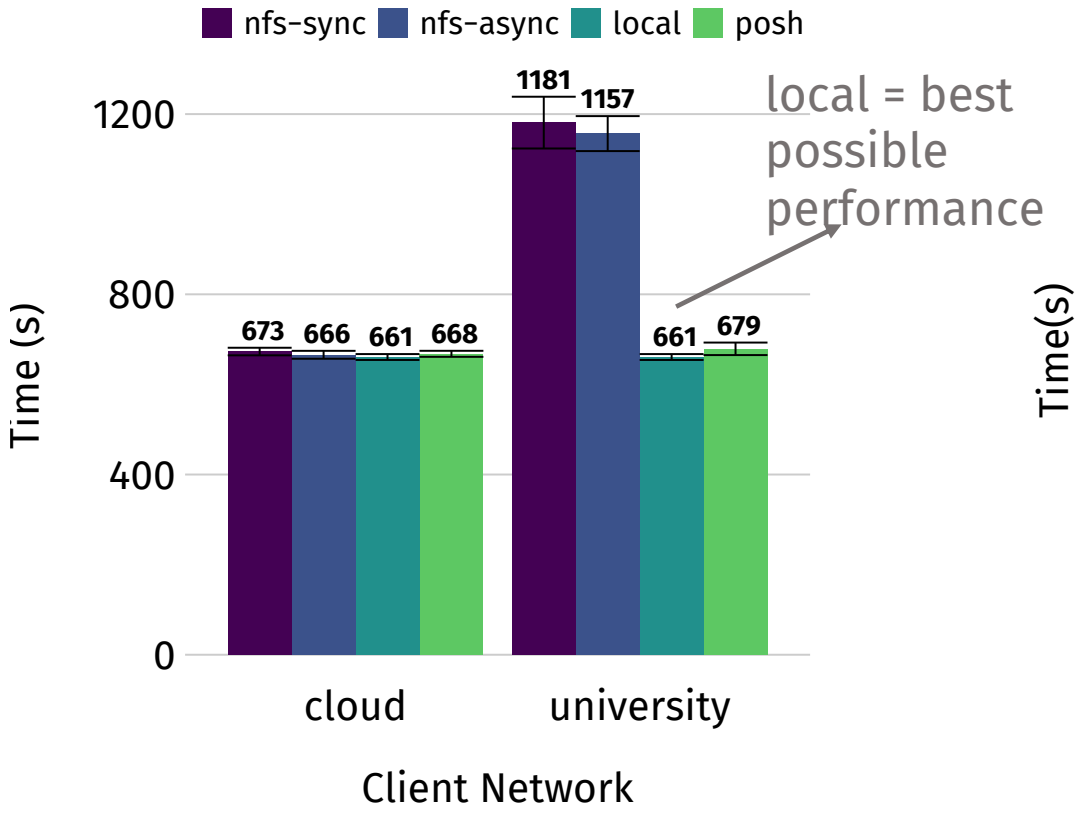
- Implementation:
  - Server component runs at proxy server that has access to NFS data
  - Client component runs at client's shell, annotations stored locally
- Evaluation:
  - Used Posh to accelerate *five unmodified shell applications*
  - Two remote NFS scenarios; proxy server runs on NFS server directly

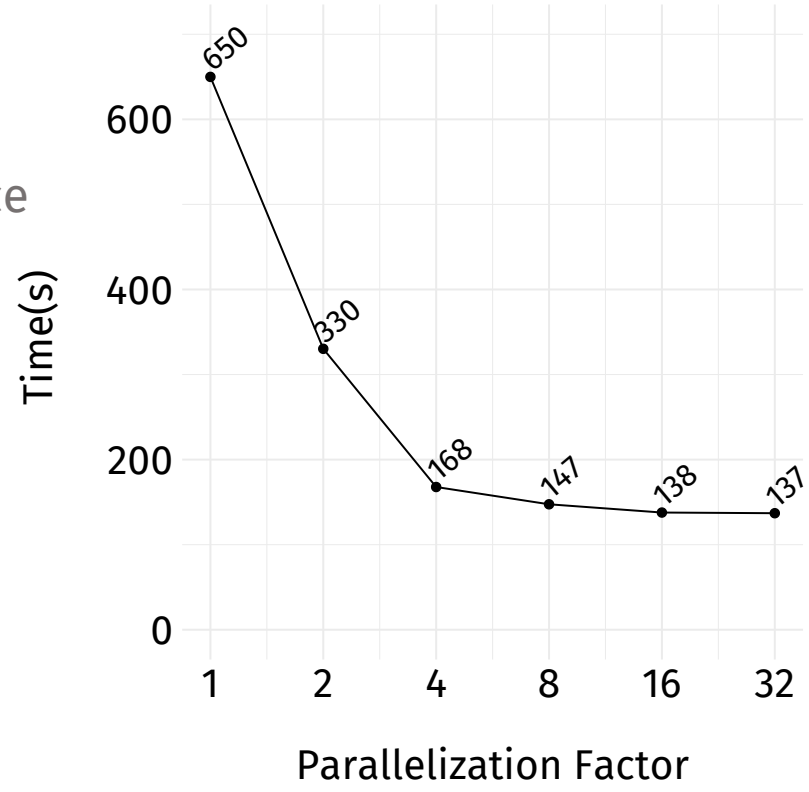
Client	Server	Throughput	RTT
Stanford	GCP Oregon	600 Mbps	20 ms
GCP Oregon	GCP Oregon	5-10 Gbps	.5 ms

Image Thumbnails with


```
mogrify -format gif -path thumbs -thumbnail 100x100 images/*.jpg
```

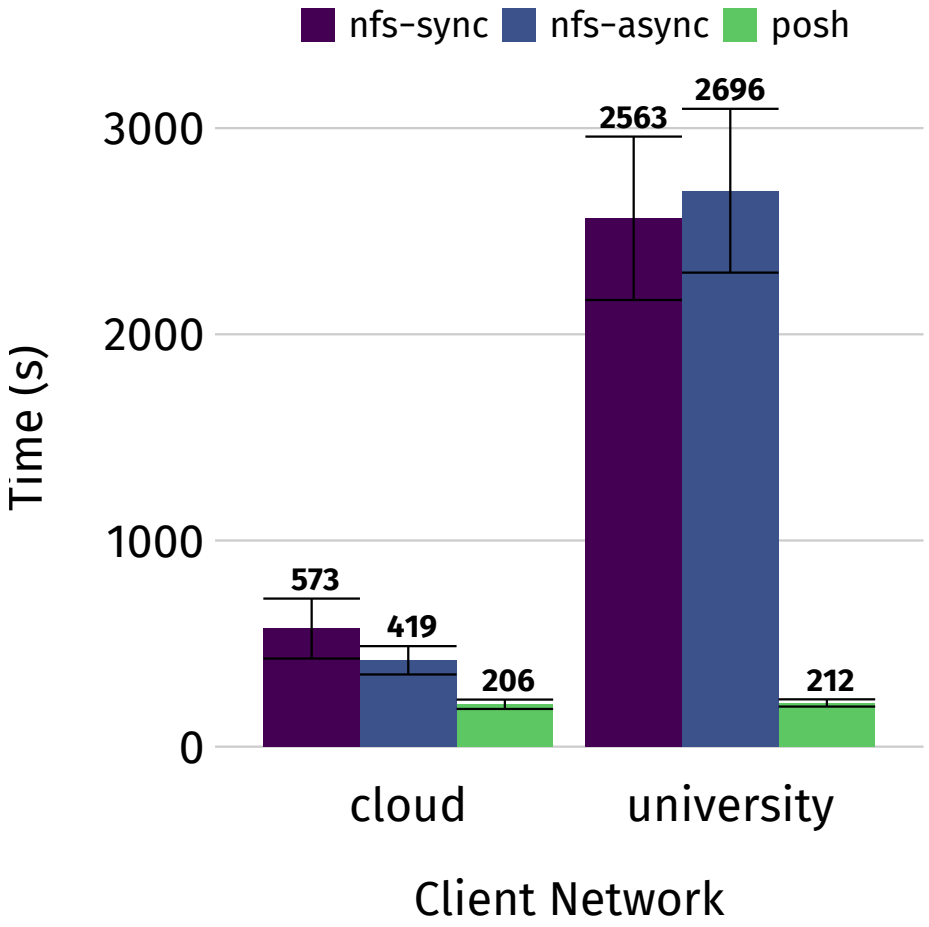
Input: 15 Gb -> Output: 12 Mb





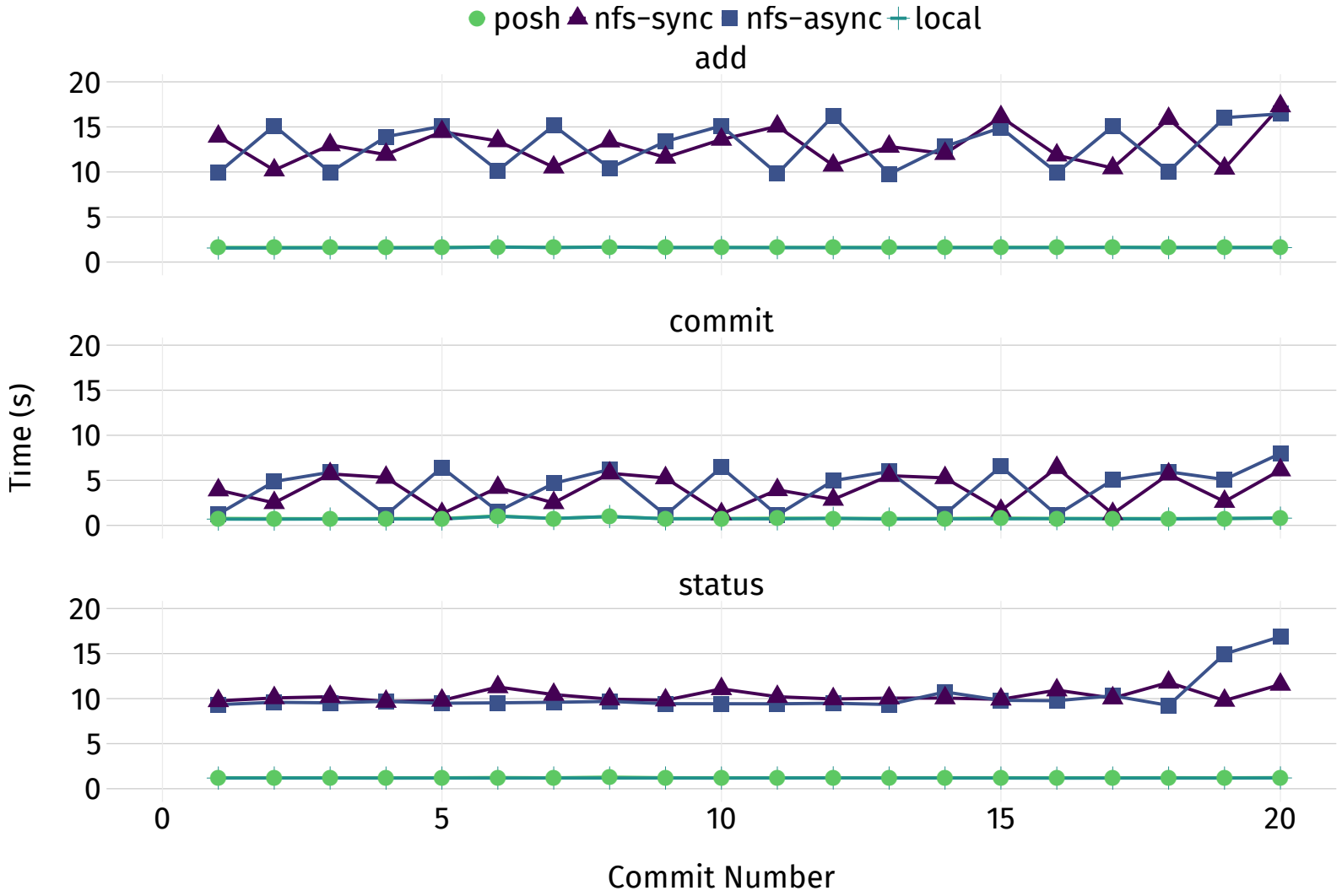
Distributed MapReduce Workload

```
cat mount0/*.csv mount1/*.csv … | grep “foo”
```



- Pipeline that runs **`cat`** and **`grep`** across files stored in *5 different remote machines*
- 80 Gb -> .8 kb
- Posh:
  - (1) offloads the command
  - (2) automatically *splits the command*
  - (3) aggregates the output in the correct order

Git Commit Workflow



- Runs **`add`**, **`commit`**, **`status`** successively across 20 chromium commits
- Two machines in same datacenter
- **`git add`** results in ~34000 to 340000 open stat calls over the network

Conclusion: shells should consider locality

- Posh introduces data locality into the shell via annotations
- Enables **1.6-15x** speedups in unmodified workloads that access remote filesystems

Contact:


deeptir@cs.stanford.edu


https://deeptir.me


github.com/deeptir18

Setup and try out Posh at: https://github.com/deeptir18/posh