The HTCondor CacheD

Derek Weitzel, Brian Bockelman
University of Nebraska — Lincoln

Today's Talk

- Today's talk summarizes work for my a part of my PhD Dissertation

- Also, this work has been accepted to PDPTA'15

HTCondor CacheD

- The CacheD puts an atomic file cache as a first class citizen in HTCondor, next to jobs.
- A cache can be scheduled…
- … be evicted…
- … have requirements…
- … stored on worker nodes…

HTCondor CacheD

- Policy language for (proactive-)replication preferences
- Ordered preferences for transfer method
  - **Bittorrent** or Condor File Transfer
- CacheD parenting.

Replication Policy Language

- HTCondor CacheD allows you to specify a policy language to match against other CacheDs.

- When creating the cache, you can say:

Replicate the cache on nodes with > X cores, and with > Y disk space

- Proactive replication can replicate to nodes while other jobs are running — saving time when your job begins.

Replication Policy Language

- On the other hand, you don't have to do proactive replication.

- Can force a cache to only be replicated when requested for a job…

Replication Transfer Method

- Each cache can have a preference for transfer method.

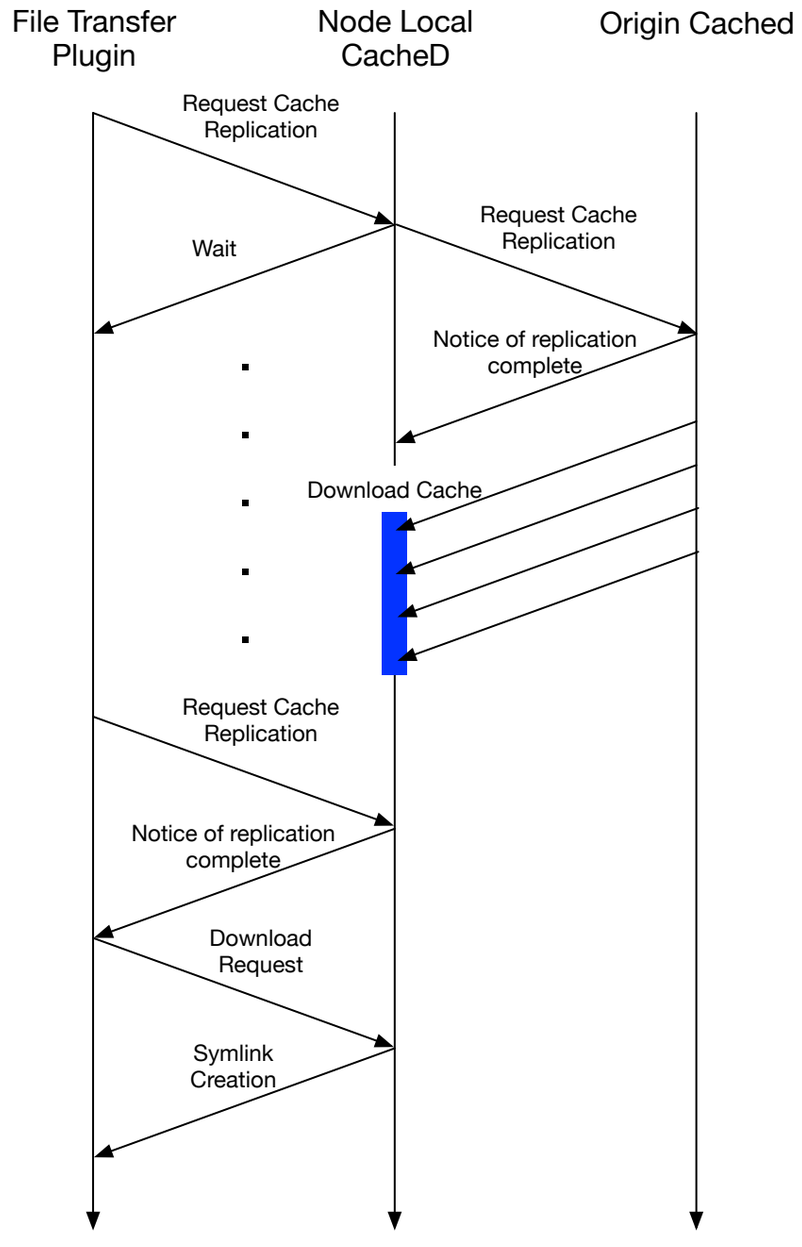
- **Direct** and **Bittorrent** are available by default.

  - Direct — Using HTCondor File Transfer. Authenticated and encrypted.

  - Bittorrent — Using Bittorrent protocol via libtorrent

Cache Protocol

1. Filetransfer requests local replication
2. Local CacheD sends request to it's parent, or origin
3. Local CacheD downloads the cache.
4. Filetransfer plugin checks status of local cache
5. Upon local cache replication, plugin downloads the cache



Once on the node

- If the job slot belongs to a CacheD, then it will symlink the cache into the execution directory.

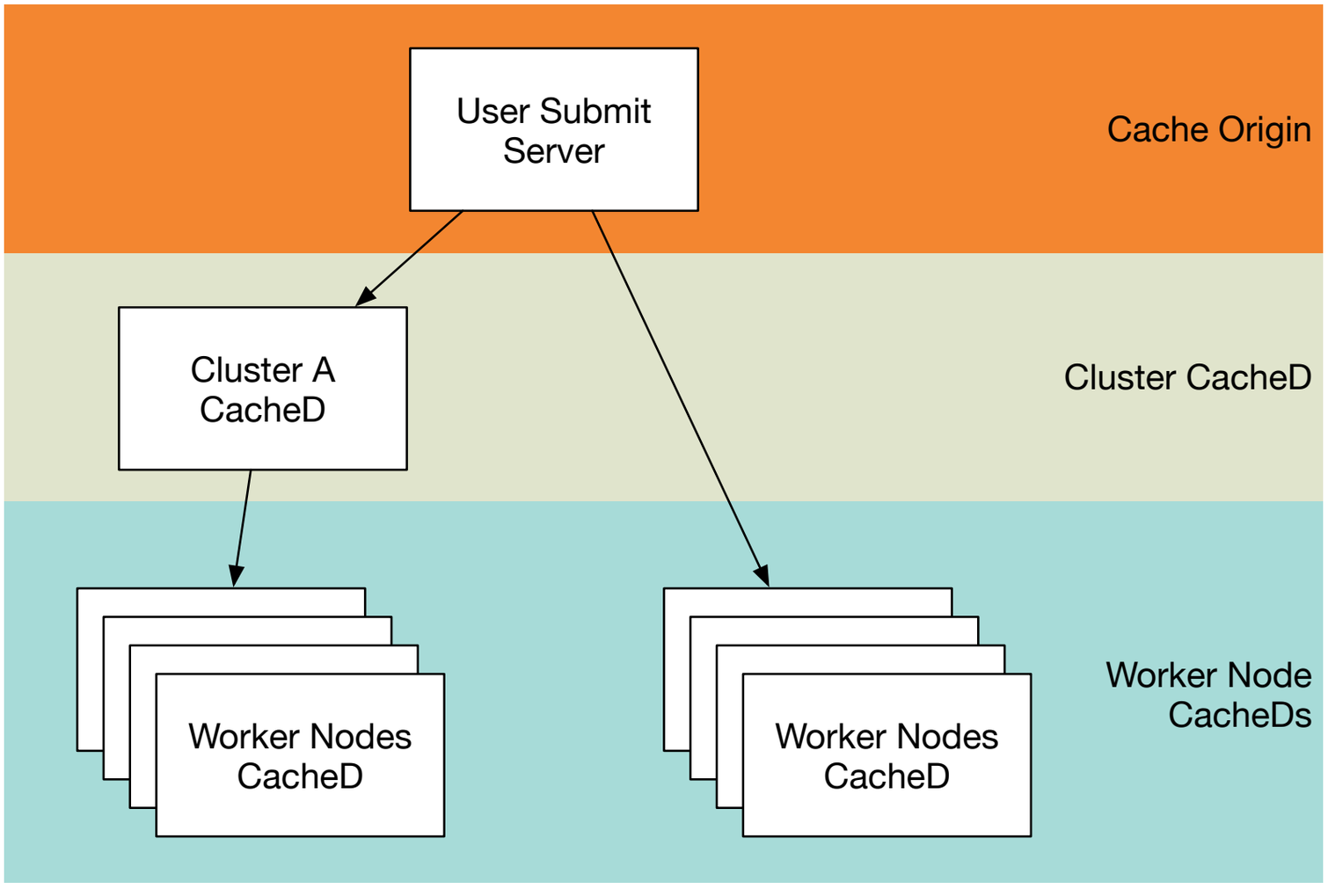
- In the glidein case, a symlink cannot be used between job slots on the same node.

  - Instead, the CacheD **Direct** copies the cache from the node local parent CacheD.

  - Slot's local CacheD copies into job's execute directory.

Parenting

- Each cache on a CacheD has a parent.
- Either it is the CacheD's parent, or the origin.



Parenting

- **Bittorrent** causes extremely high IO load on a node.

- Every CacheD can parent to a node local CacheD so only one will be using Bittorrent at a time.

- All children CacheD's will copy the cache with **Direct** transfer method.

Evaluation

- Since this is for my PhD on campus computing, we first evaluated the CacheD against a local campus cluster.
- Submitted a BLAST benchmark using the NR database.
- Looked at Stage-in without cache, and with cache.

Stage-in — No Cache

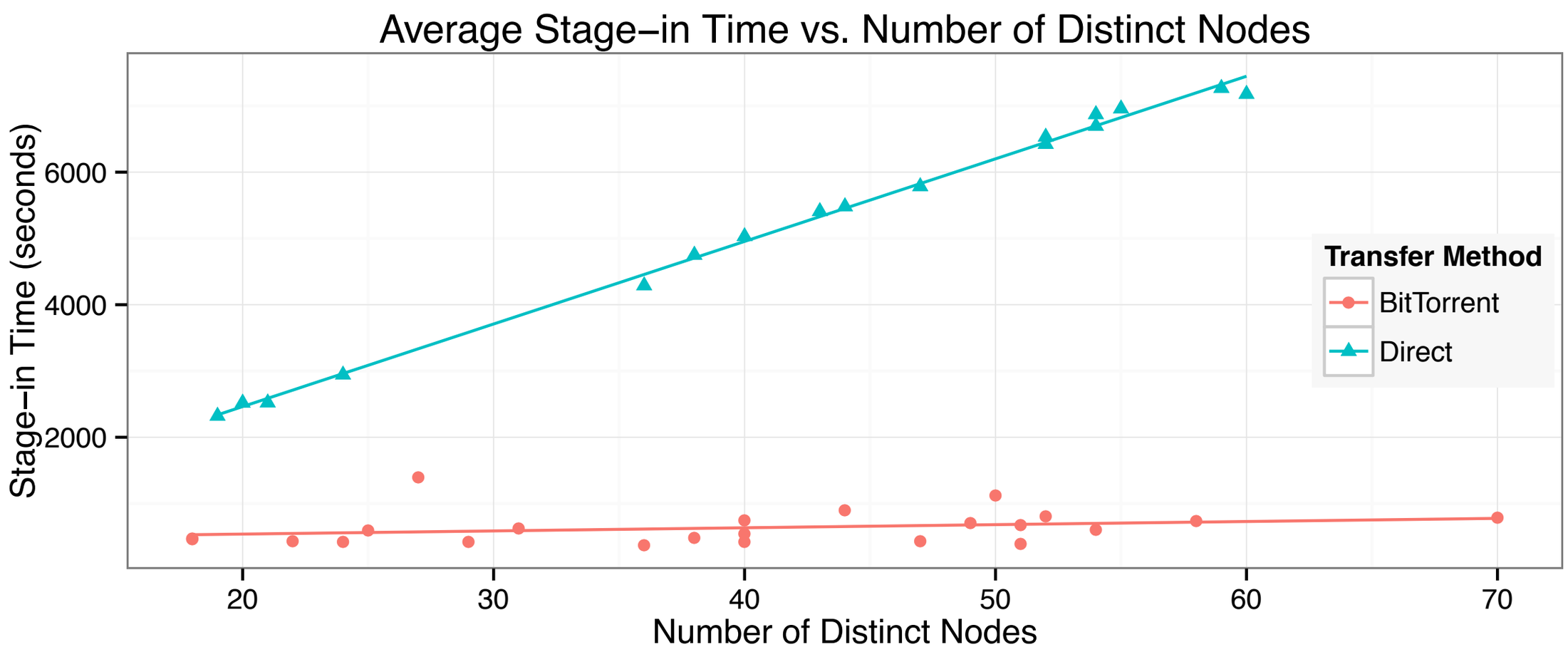
- Measures the transfer speed of **Bittorrent** vs. **Direct** HTCondor file transfers.

- Evaluated against number of distinct nodes downloading a copy of the cache.

- Origin server for the tests has a 1Gbps connection.

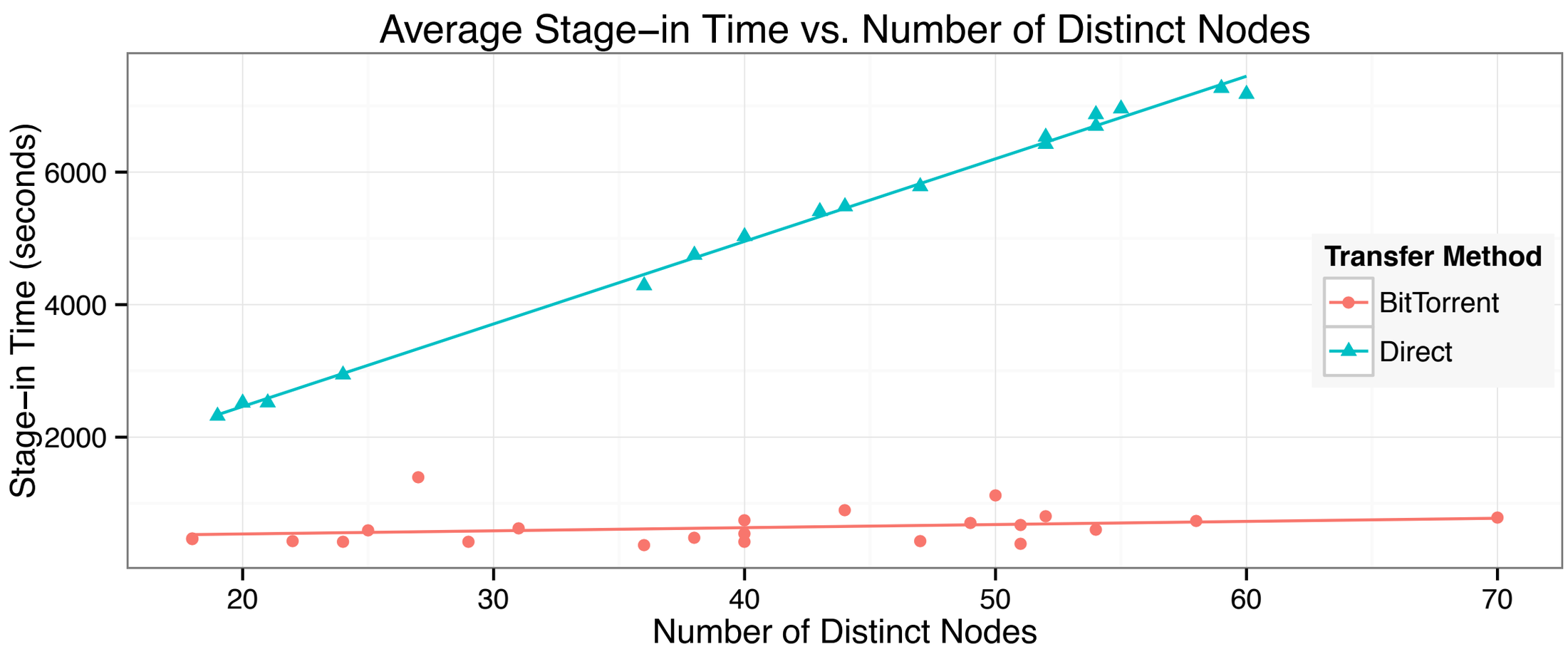
Stage-in — No Cache

- Measured time required to stage-in 15GB of the NR database vs. number of distinct nodes.



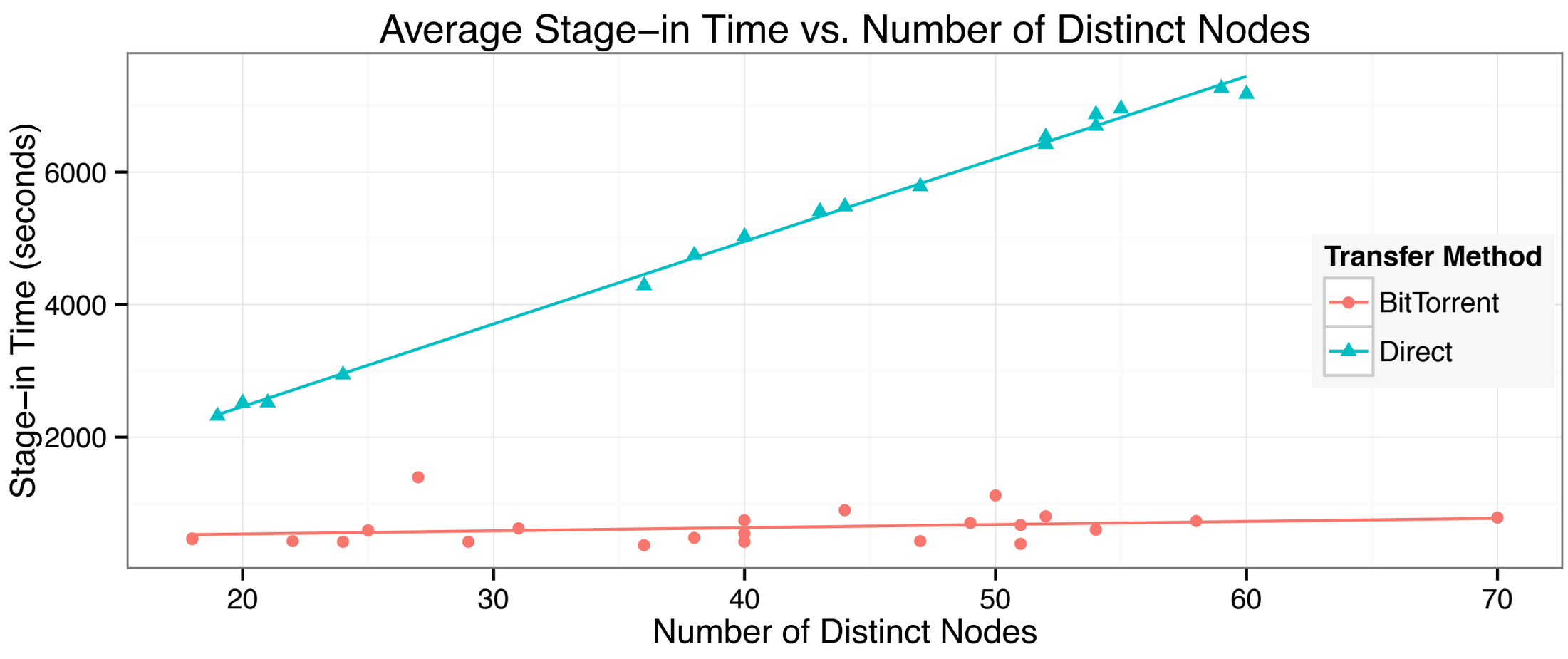
Stage-in — No Cache

- The Direct method had a linear increase in average stage-in time.



Stage-in — No Cache

- Bittorrent had very small, if any, increase in transfer time as the number of distinct nodes increases.

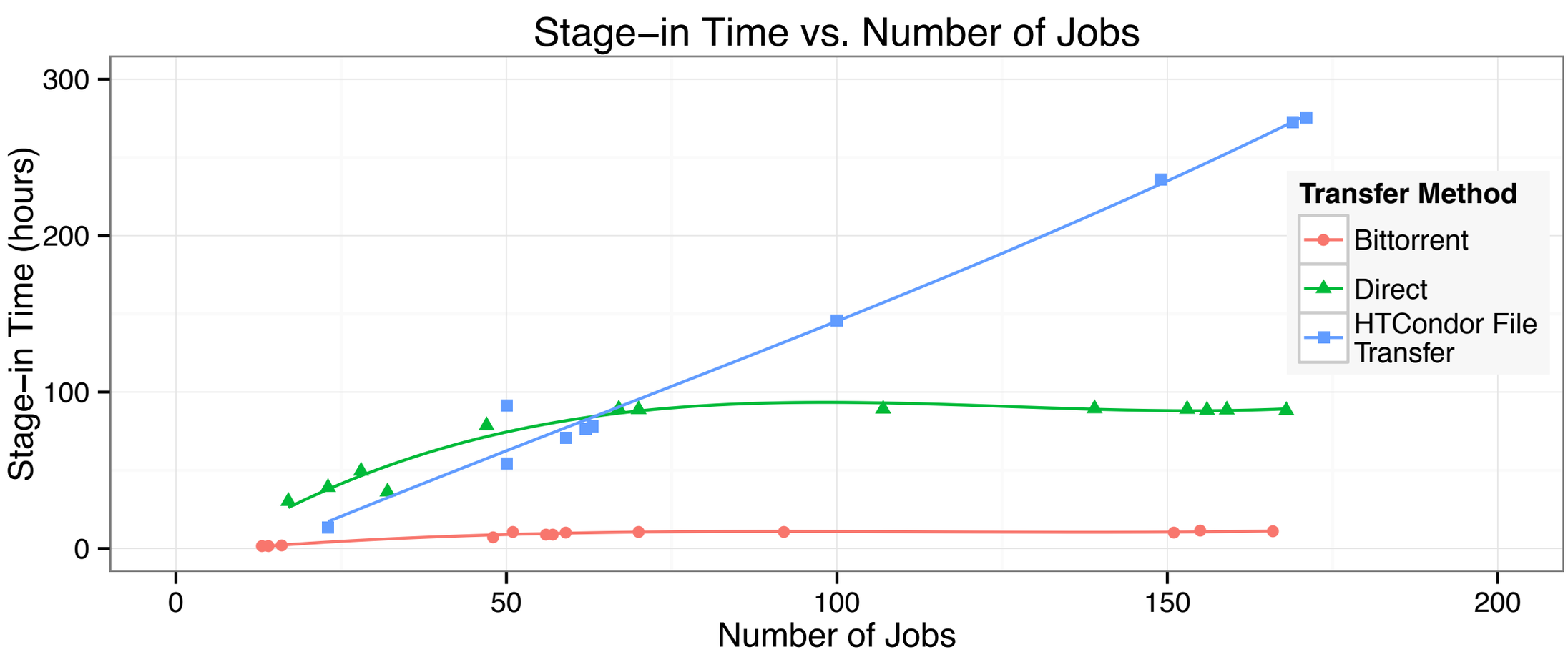


Stage-in — Cached

- Ran varying number of jobs with same number of available nodes (50).
- Measure the total stage-in time for the cache.
- Once the cache is stored on the node, replications are quick.

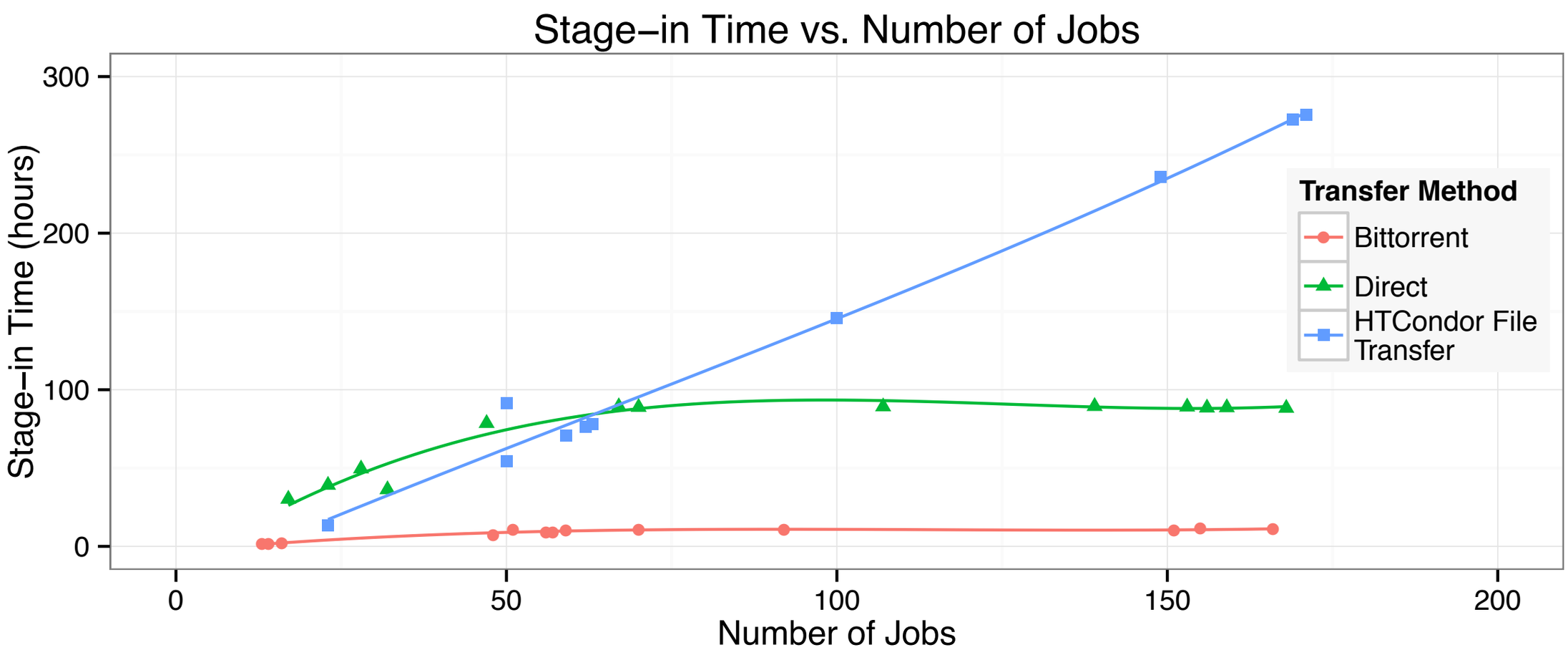
Stage-in — Cached

- Noticeable curve at 50 — subsequent jobs have 0 stage-in time.



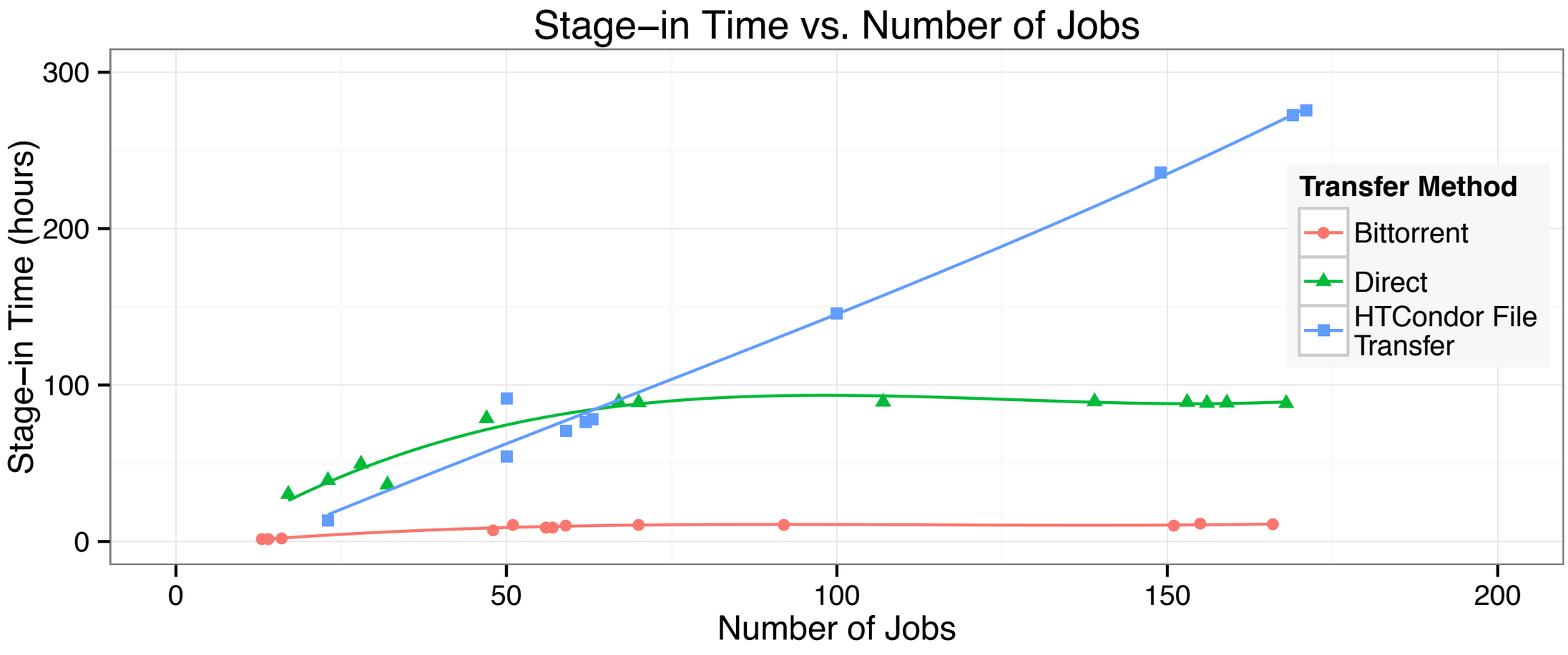
Stage-in — Cached

- Again, **Bittorrent** is faster than **Direct** and HTCondor file transfer method.



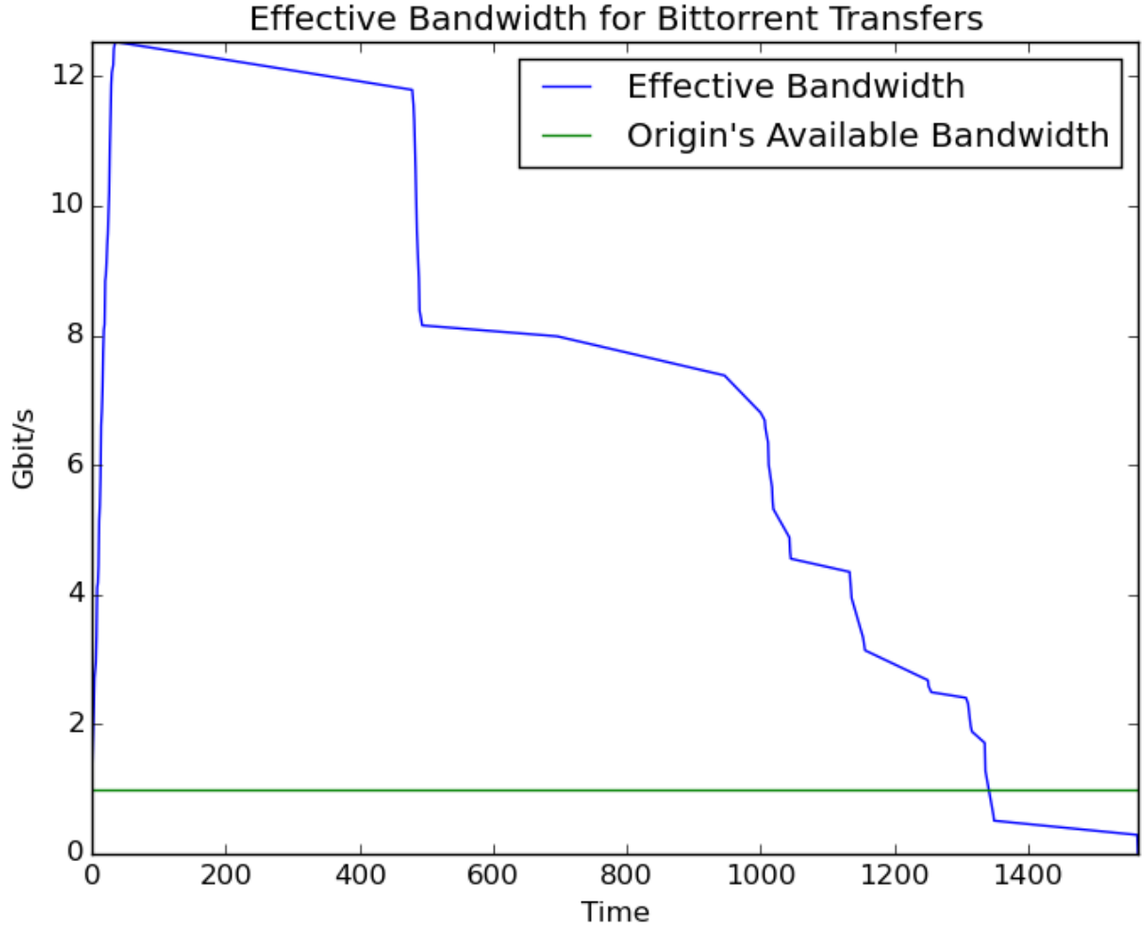
Stage-in — Cached

- Caching significantly decreases stage-in time when the there is a cache hit.



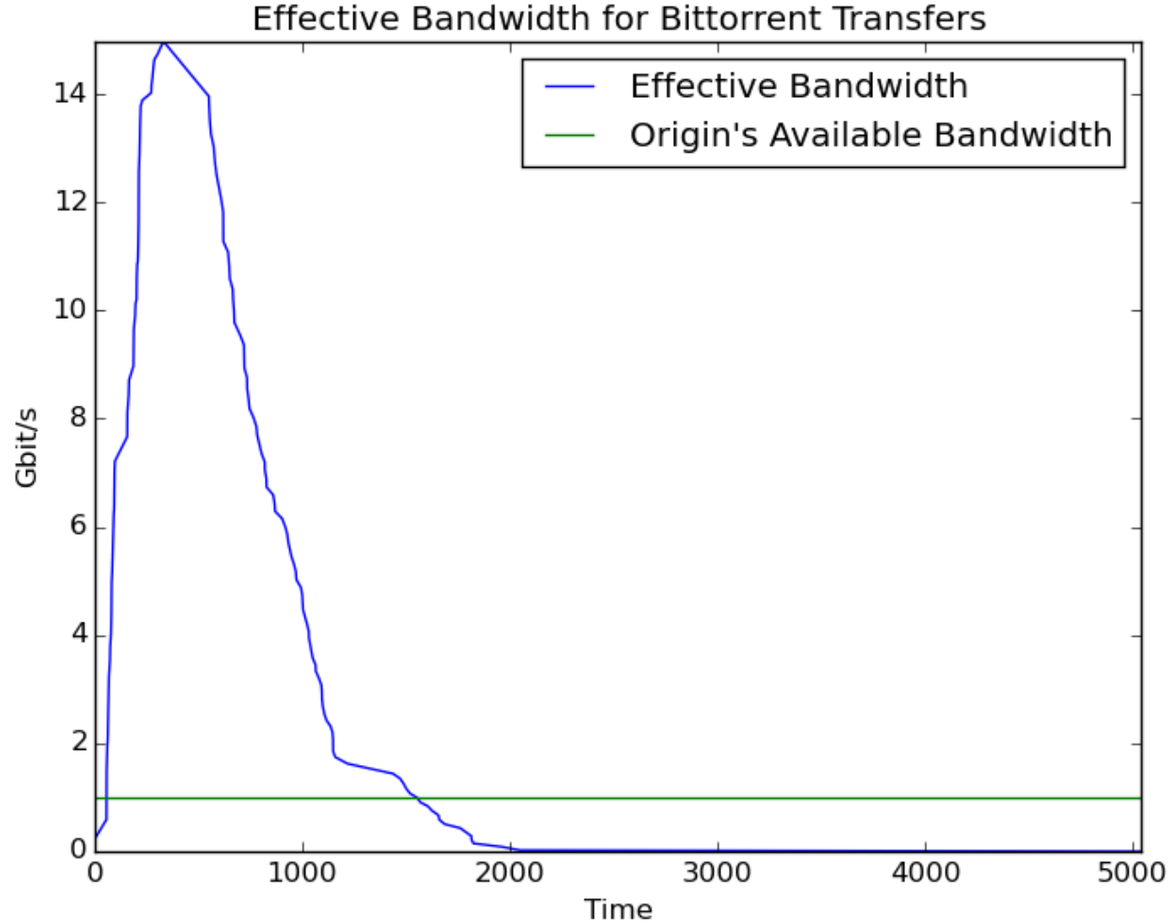
Additional Observations

- **Bittorrent** multiplies the available bandwidth to a cache origin server.



Additional Observations

- **Bittorrent** multiplies the available bandwidth to a cache origin server — Even on the OSG



Conclusions

- **Bittorrent** is a very efficient method for transferring common files on a cluster.

- The CacheD is a flexible agent to store common files for reuse.

- Currently running tests on the effectiveness of using the CacheD on the Open Science Grid.