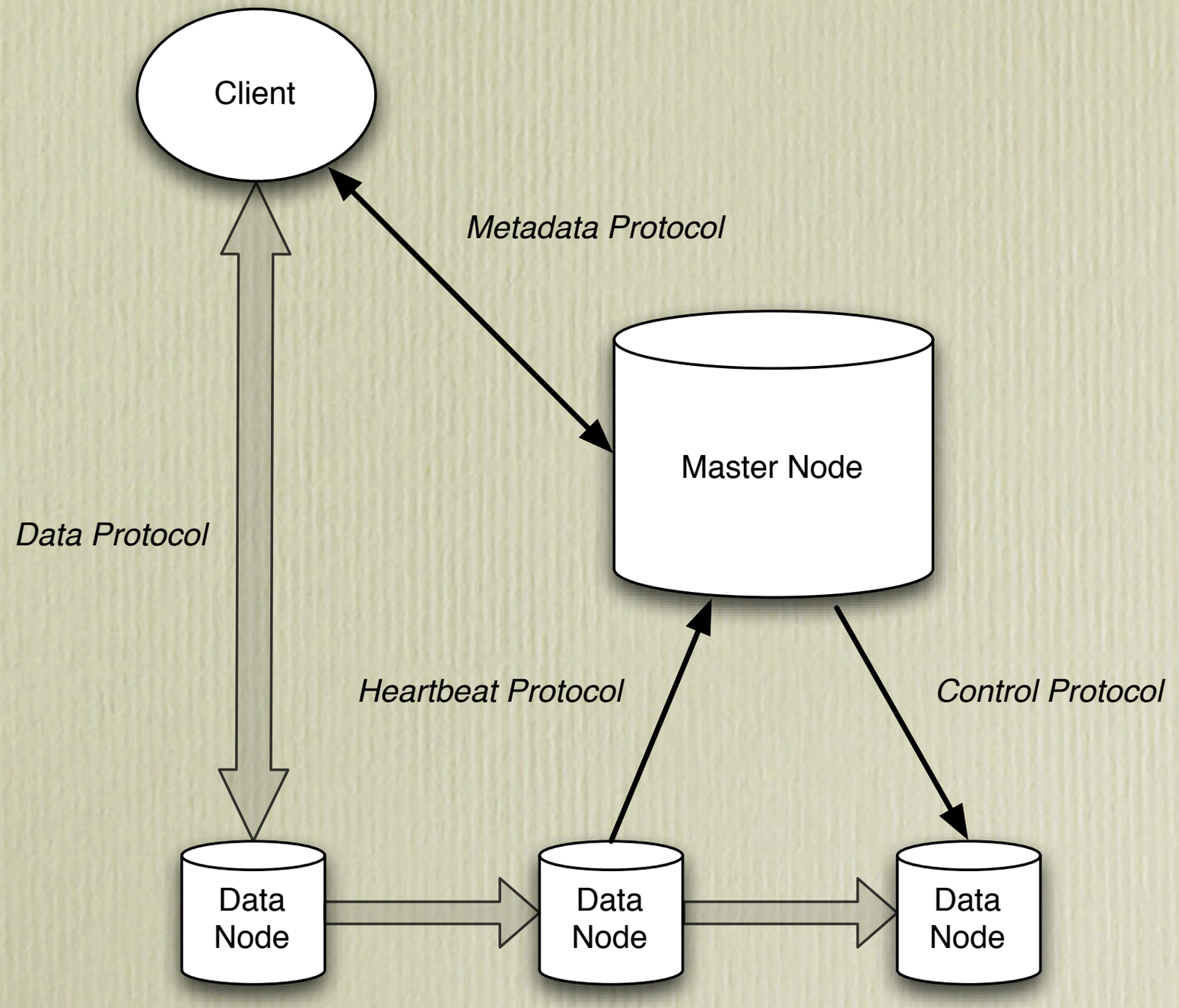
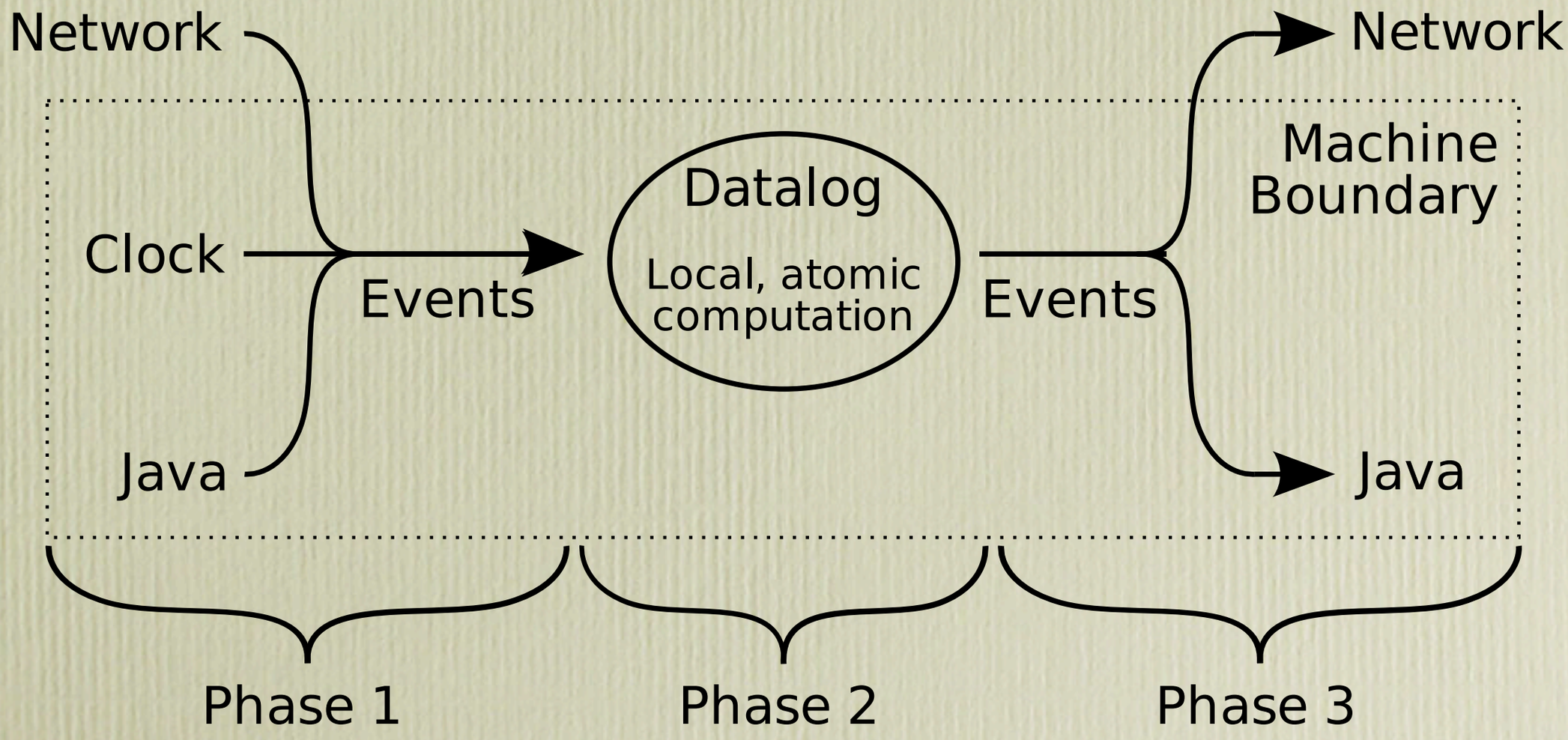
BOOM: Data-Centric Programming For the Data Center

Peter Alvaro and Neil Conway
UC Berkeley DB Group

HDFS Architecture



An Overlog Timestep



Simple Request/Response

```
request(@Master, Client, ReqType, Args) :-
    start_request(@Client, Master, ReqType, Args);
```

Simple Request/Response

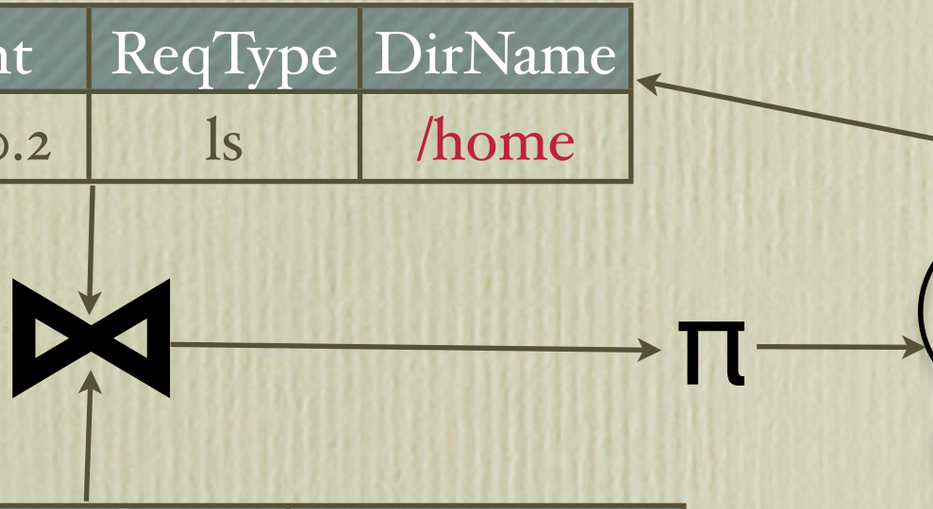
```
request(@Master, Client, ReqType, Args) :-
    start_request(@Client, Master, ReqType, Args);

response(@Client, true, DirContents) :-
    request(@Master, Client, ReqType, DirName),
    ReqType == “ls”,
    directory(@Master, DirName, DirContents);
```

Example Data Flow

```
response(@Client, true, DirContents) :-
    request(@Master, Client, ReqType, DirName),
    ReqType == "ls",
    directory(@Master, DirName, DirContents);
```

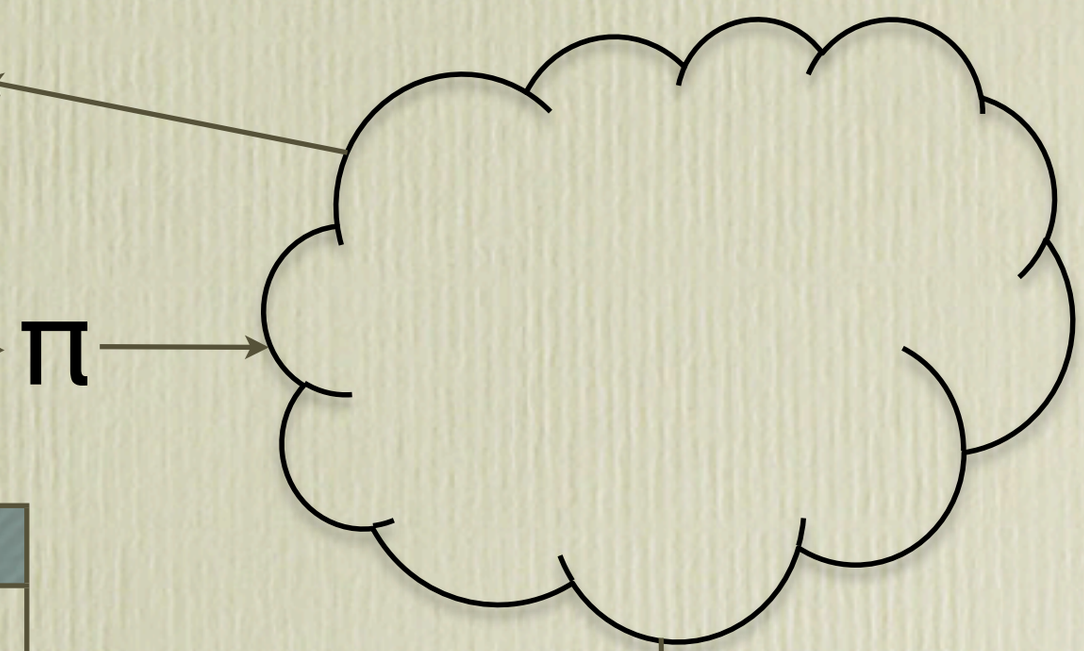
request

@Master	Client	ReqType	DirName
10.0.0.1	10.0.0.2	ls	/home



directory

@Master	Dirname	DirContents
10.0.0.1	/	{ home, tmp }
10.0.0.1	/home	{ foo, bar }



response

@Client	Success	DirContents
10.0.0.2	TRUE	{ foo, bar }

File System State

- file(Master, FileId, FName, FParentId, IsDir)
- chunk(Master, ChunkId, FileId)
- fqpath(Master, Path, FileId)

Example: “ls”

```
response(@Client, true, DirContents) :-
    request(@Master, Client, ReqType, DirName),
    ReqType == “ls”,
    directory(@Master, DirName, DirContents);

response(@Client, false, null) :-
    request(@Master, Client, ReqType, DirName),
    ReqType == “ls”,
    notin directory(@Master, DirName, _);

directory(@Master, DirName, set<FileName>) :-
    fqpath(@Master, DirName, DirId),
    file(@Master, FileId, DirId, FileName, _);
```

Disjunction

Aggregation

State Updates

```
file(@Master, FileId, DirName, FParentId, true) :-
    request(@Master, _, ReqType, {Parent, DirName}),
    fqpath(@Master, Parent, FParentId),
    ReqType == “mkdir”,
    FileId := newFileId();

response(@Client, true, null) :-
    request(@Master, Client, ReqType, {Parent, _}),
    fqpath(@Master, Parent, _),
    ReqType == “mkdir”;

response(@Client, false, null) :-
    request(@Master, Client, ReqType, {Parent, _}),
    notin fqpath(@Master, Parent, _),
    ReqType == “mkdir”;
```

Heartbeats and Timers

```
timer(hb_clock, 1000);

heartbeat(@Master, Datanode, Tstamp, ChunkId) :-
      dn_chunk(@Datanode, ChunkId),
      master(@Datanode, Master),
      hb_clock(@Datanode, Tstamp);
```

Heartbeats and Timers

```
timer(hb_clock, 1000);

heartbeat(@Master, Datanode, Tstamp, ChunkId) :-
    dn_chunk(@Datanode, ChunkId),
    master(@Datanode, Master),
    hb_clock(@Datanode, Tstamp);


hb_cache(@Master, Datanode, Tstamp) :-
    heartbeat(@Master, Datanode, Tstamp, _);

hb_chunk(@Master, Datanode, ChunkId) :-
    heartbeat(@Master, Datanode, _, ChunkId);
```

Invariant Maintenance

```
clone_chunk(@Source, Target, ChunkId) :-
    exemplar(@Master, ChunkId, Source),
    candidate(@Master, ChunkId, Target),
    chunk_cnt(@Master, ChunkId, Cnt),
    Cnt < 3;

chunk_cnt(@Master, ChunkId, count<Datanode>) :-
    heartbeat_cache(@Master, Datanode, _),
    dn_chunk(@Master, Datanode, ChunkId);

delete
hb_cache(@Master, Datanode, Tstamp) :-
    hb_cache(@Master, Datanode, Tstamp),
    time() - Tstamp > 5000;
```

Data Path

Does It Work?

Does It Work?

- How fast is it?
  - Within ~20% of HDFS for Hadoop

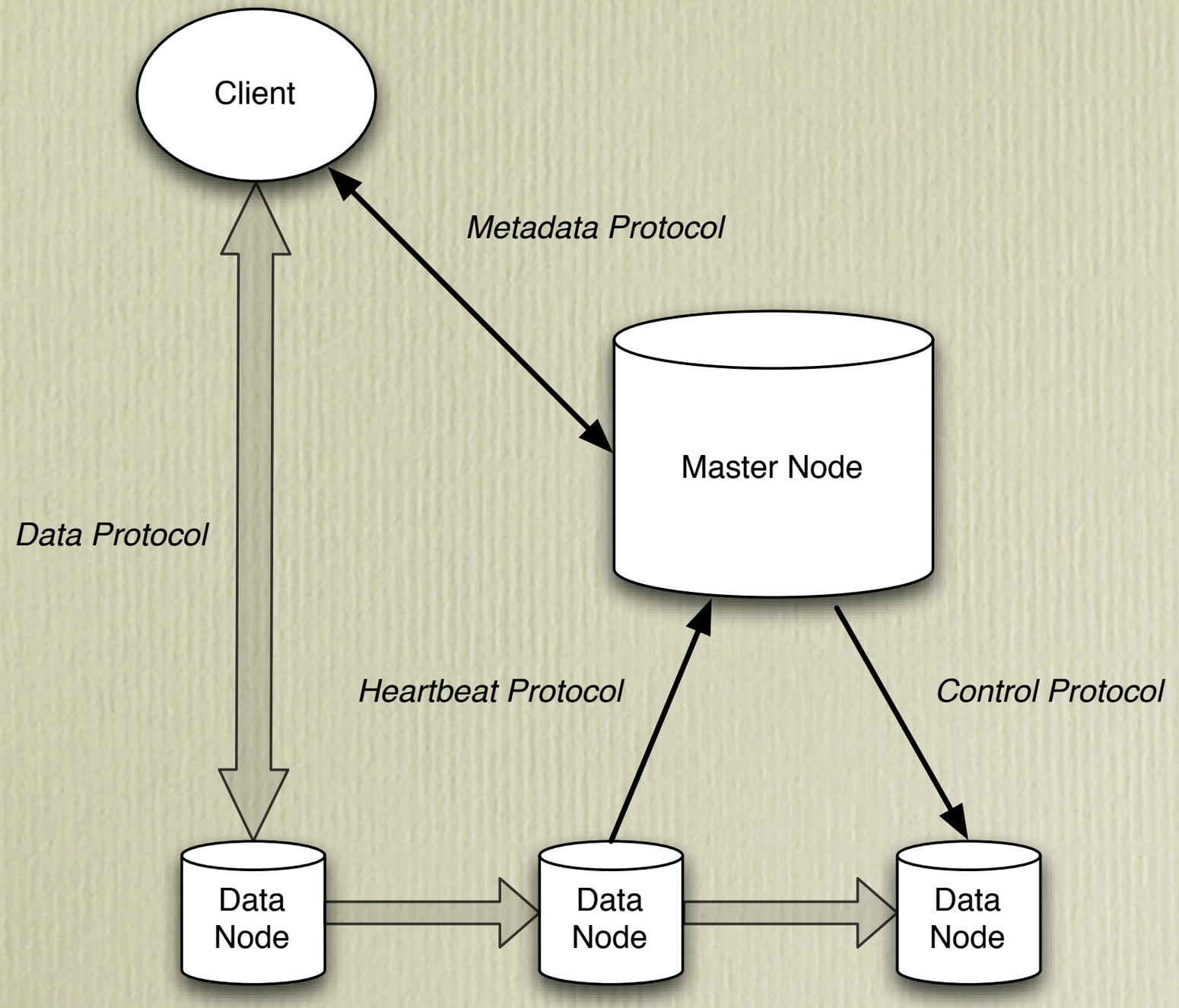
Does It Work?

- How fast is it?
  - Within ~20% of HDFS for Hadoop
- How complex is it?
  - BoomFS: 469 LOC Overlog, 1431 LOC Java
  - HDFS: 21,700 LOC Java

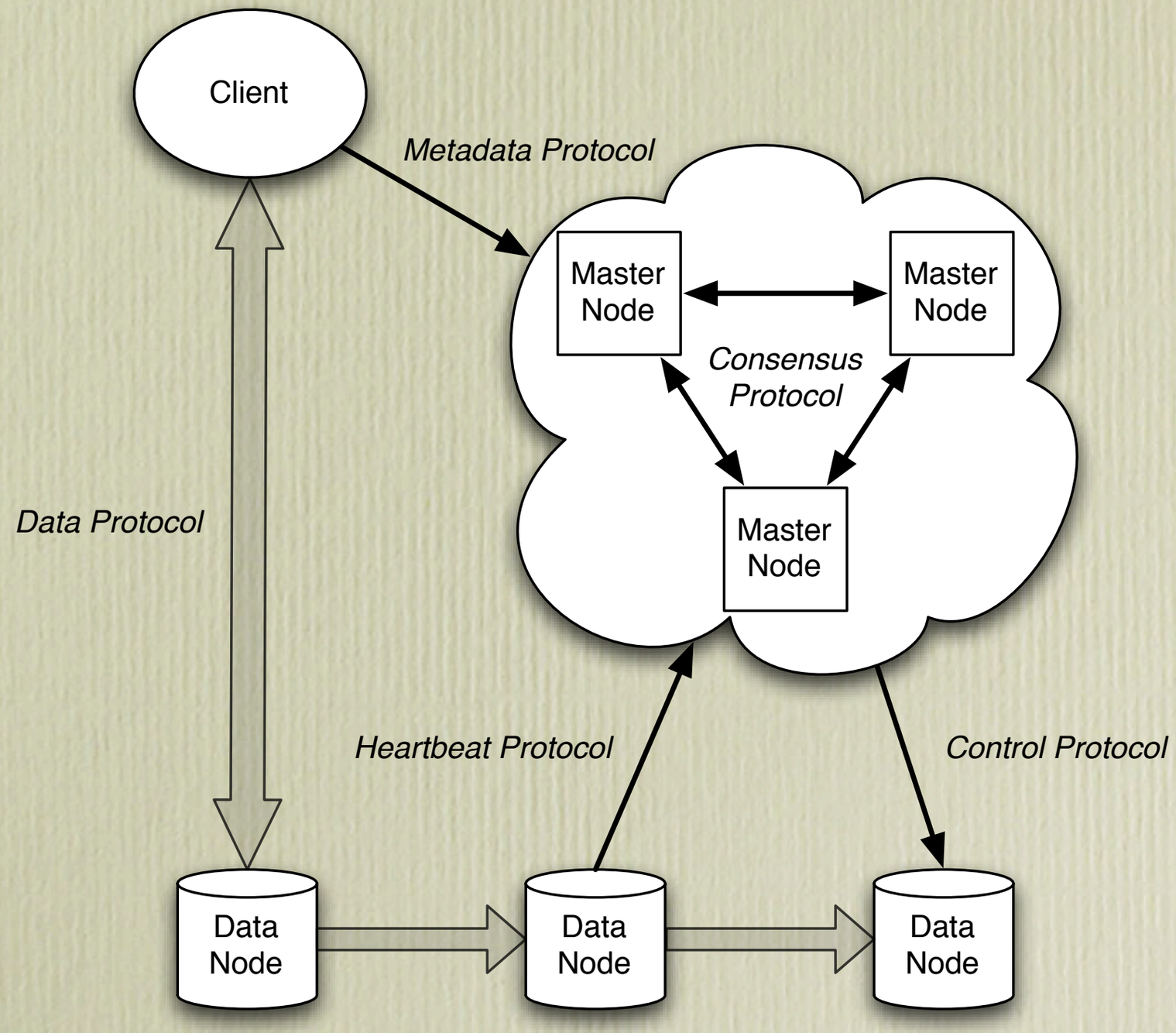
Does It Work?

- How fast is it?
  - Within ~20% of HDFS for Hadoop
- How complex is it?
  - BoomFS: 469 LOC Overlog, 1431 LOC Java
  - HDFS: 21,700 LOC Java
- How easy is it to change?

HDFS Architecture



Architecture with Consensus



State Updates

```
file(@Master, FileId, DirName, FParentId, true) :-
    request(@Master, _, “mkdir”, {Parent, DirName}),
    fqpath(@Master, Parent, FParentId),
    FileId := newFileId();

response(@Client, true, null) :-
    request(@Master, Client, “mkdir”, {Parent, _}),
    fqpath(@Master, Parent, _);
```

Consensus

```
pending(@Master, Parent, DirName, “mkdir”) :-
    request(@Master, _, “mkdir”, {Parent, DirName});

file(@Self, FileId, FName, FParentId, true) :-
    passed(@Self, Parent, DirName, “mkdir”),
    fqpath(@Self, Parent, FParentId),
    FileId := newFileId();

response(@Client, true, null) :-
    passed(@Master, Parent, DirName, “mkdir”),
    request(@Master, Client, “mkdir”, {Parent, DirName}),
    fqpath(@Master, Parent, _);
```

Interposition

Manageability

```
r_mkdir
file(@Master, FileId, DirName, FParentId, true) :-
    request(@Master, _, “mkdir”, {Parent, DirName}),
    fqpath(@Master, Parent, FParentId),
    FileId := newFileId();
```

```
tap(@Master, “r_mkdir”, Tstamp, count<*>) :-
    request(@Master, _, “mkdir”, {Parent, DirName}),
    fqpath(@Master, Parent, FParentId),
    Tstamp := time();
```

Next Steps

Filesystem:

- Deploy BoomFS in production
- Erasure codes, routing policies, ...

BOOM Stack:

- Declarative Hadoop (mostly finished)
- Group communication, persistent queues
- Cloud storage across data centers (e.g. Sherpa)

Language:

- Host-language integration, syntax sugar
- Multi-core evaluation
- Query optimization for distributed systems

Thanks!

BOOM Team:

Peter Alvaro

Tyson Condie

Joe Hellerstein

Neil Conway

Russell Sears

Feedback:
nrc@cs.berkeley.edu, palvaro@cs.berkeley.edu

Fully-qualified Paths

Fully-qualified Paths

```
fqpath(@Master, Path, FileId) :-
    file(@Master, FileId, FParentId, _, true),
    FParentId == null,
    Path := “/”;
```

Base Case

```
fqpath(@Master, Path, FileId) :-
    file(@Master, FileId, FParentId, FName, _),
    fqpath(@Master, ParentPath, FParentId),
    PathSep := (ParentPath == “/” ? “” : “/”),
    Path := ParentPath + PathSep + FName;
```

Inductive Case