

DynamiteDB

Satyajeet Gawas

Yakshdeep Kaul

Tyler Lisowski















Ramon Sua

Overall Design



System in AWS!

“Elastic” IP

<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP
<input type="checkbox"/>	DB Nodes	i-e271d87f	t2.micro	us-east-1a	 running	✓ 2/2 checks ...	None	 ec2-52-200-241-247.co...	52.200.241.247
<input type="checkbox"/>		i-5171d8cc	t2.micro	us-east-1a	 running	✓ 2/2 checks ...	None	 ec2-52-200-248-223.co...	52.200.248.223
<input type="checkbox"/>		i-5071d8cd	t2.micro	us-east-1a	 running	✓ 2/2 checks ...	None	 ec2-52-200-254-246.co...	52.200.254.246
<input type="checkbox"/>		i-5371d8ce	t2.micro	us-east-1a	 running	✓ 2/2 checks ...	None	 ec2-52-201-0-131.com...	52.201.0.131
<input type="checkbox"/>		i-1a913887	t2.micro	us-east-1a	 running	✓ 2/2 checks ...	None	 ec2-52-200-255-102.co...	52.200.255.102
<input type="checkbox"/>	Master Nodes	i-c8766d31	t2.micro	us-east-1e	 running	✓ 2/2 checks ...	None	 ec2-52-201-211-111.co...	52.201.211.111
<input type="checkbox"/>		i-46524bbf	t2.micro	us-east-1e	 running	✓ 2/2 checks ...	None	 ec2-54-173-24-118.co...	54.173.24.118

- Custom Ubuntu Images made for both types of nodes
- Security groups used to limit inbound traffic on nodes
- Git + Maven used to automate source download & build process

Master and Client

Timestamp based Conflict Resolution:

VC_dbnode1=([A,2],[B,1])

VC_dbnode2=([A,1],[B,2])

CONFLICT!!!

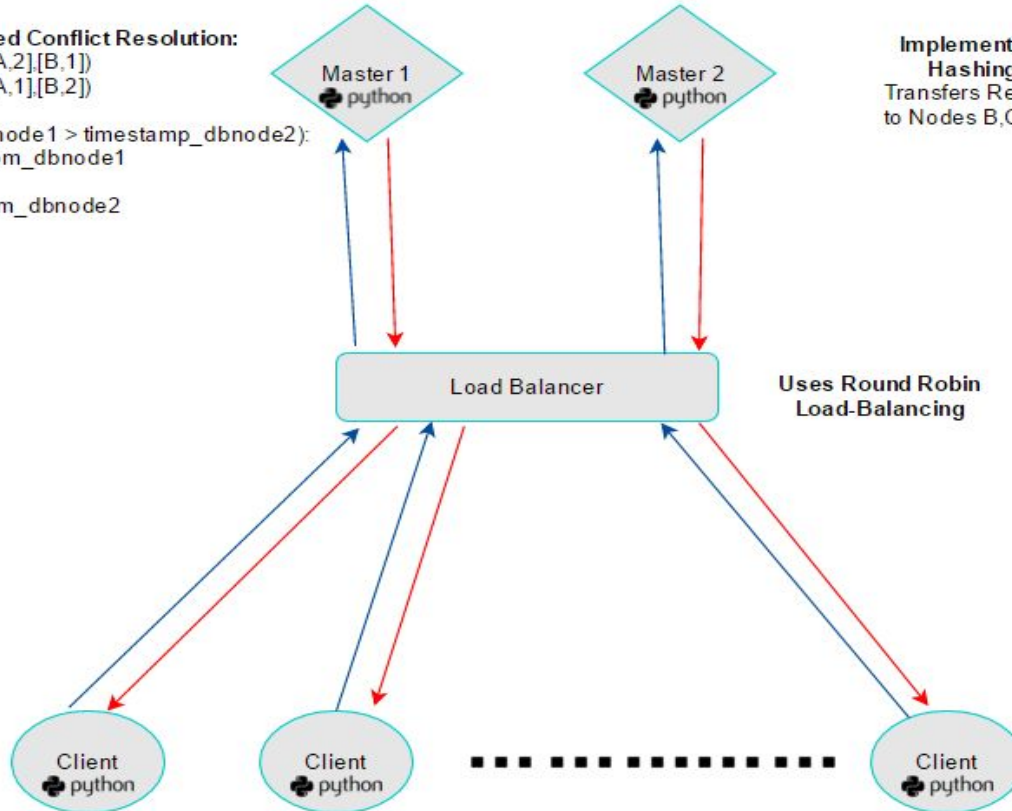
if(timestamp_dbnode1 > timestamp_dbnode2):

return reply_from_dbnode1

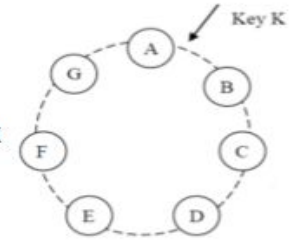
else

return reply_from_dbnode2

Threaded Implementation &
Handles dbnode Failure



Implements Consistent
Hashing Protocol:
Transfers Request for Key K
to Nodes B,C,D



→ Request

→ Reply

GET/PUT Request:

data={}

data[METHOD]=GET/PUT

data[KEY]=dummy_key

data[VALUE]=dummy_val

DB Nodes- Consistent Hashing & Replication



***For our N=3 design!**

DB Nodes- Anti-Entropy Background Process



↖ JSON Format

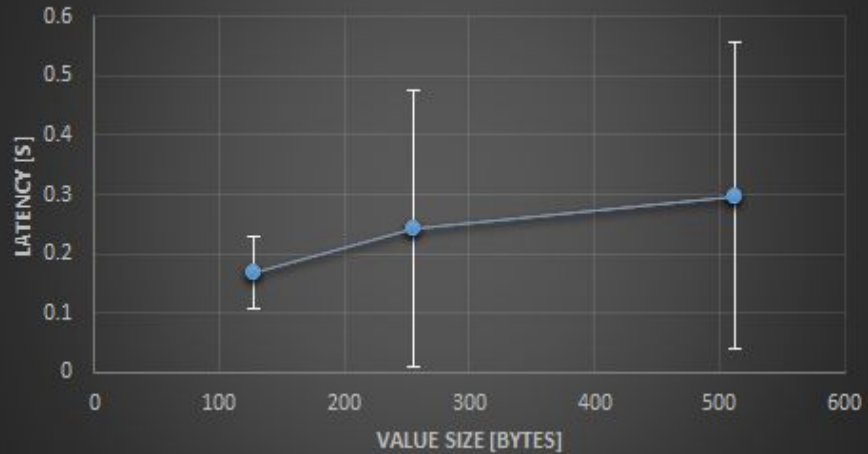
Demo

Performance Evaluation

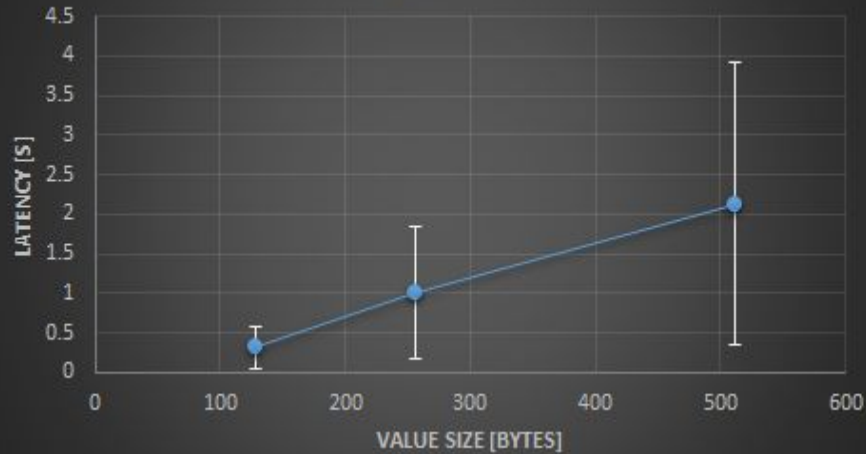
- GET and PUT request latency
- Anti-Entropy Time
- Distribution of keys across database nodes

Results

'GET' Request Latency



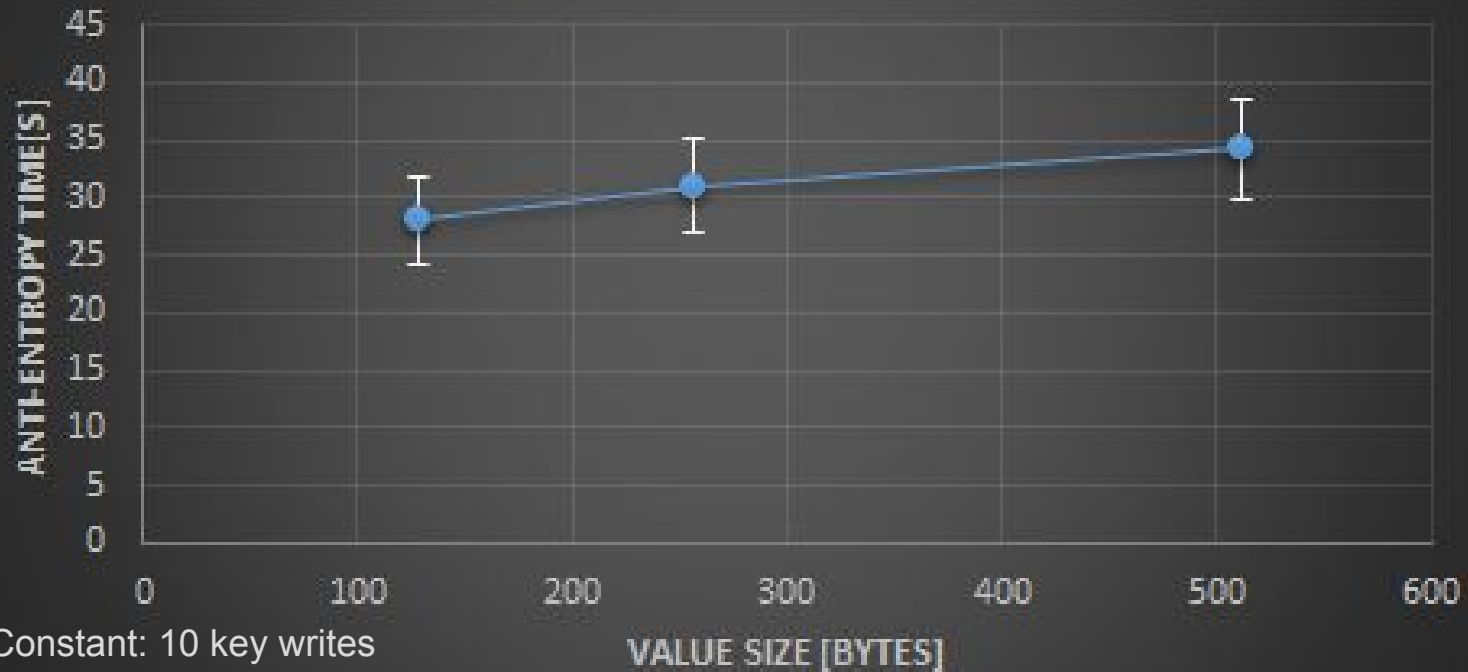
'PUT' Request Latency



Trials: 10x

Anti-Entropy Time

Varying Value Size

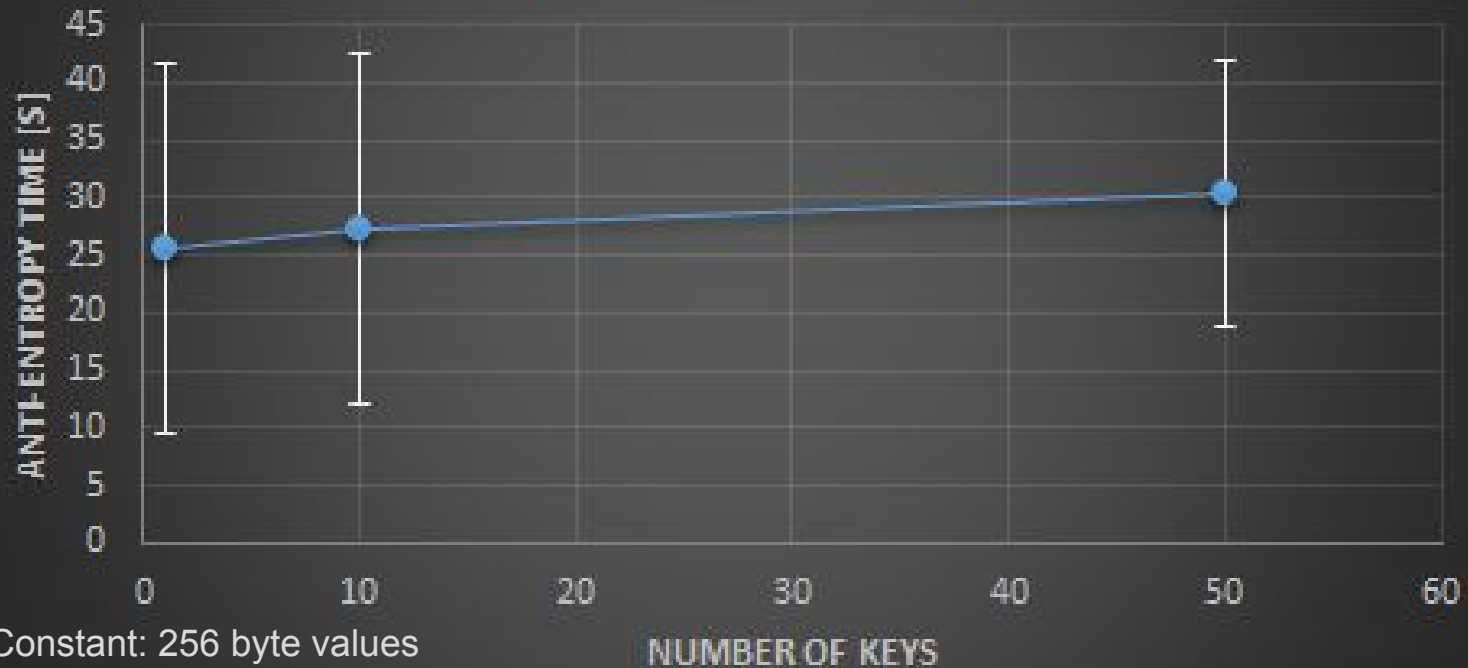


Constant: 10 key writes

Trials: 10x

Anti-Entropy Time

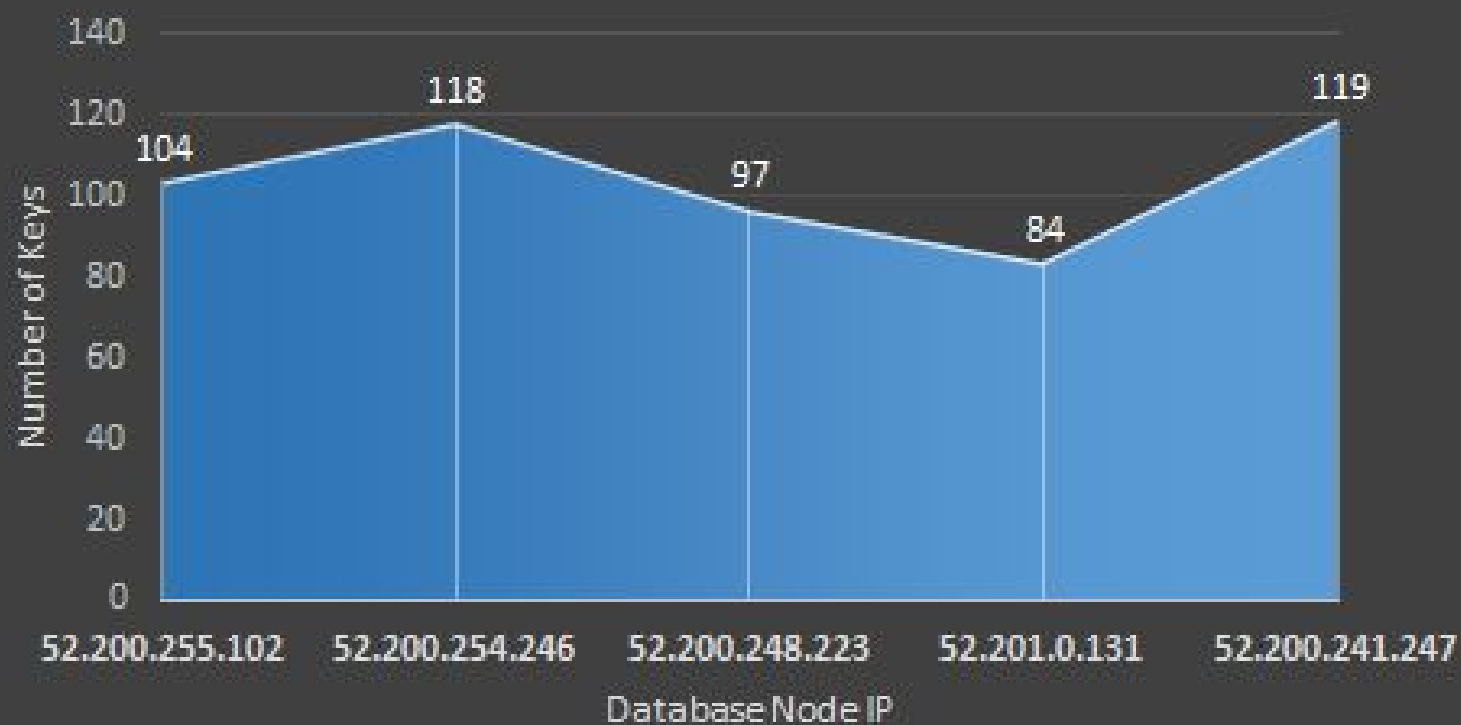
Varying Number of Keys



Constant: 256 byte values

Trials: 10x

Load Distribution



Future Work

- Merkle trees → Less keys exchanged
- Dynamic node addition/removal → More scalability
- No global routing knowledge → Less memory for routing table
- Cache key-value objects → Improved request latency

QUESTIONS?