A Case for Continuation in Servers

Hiroshi Yamauchi
Java Platform Group, Google

Talk Outline

Problem: “Sync vs Async Dilemma”

Solution: Continuation Server

Experimental results

Conclusion

Server


Client
Request
Frontend Server
RPC
Backend Server (DB)
Response
RPC return

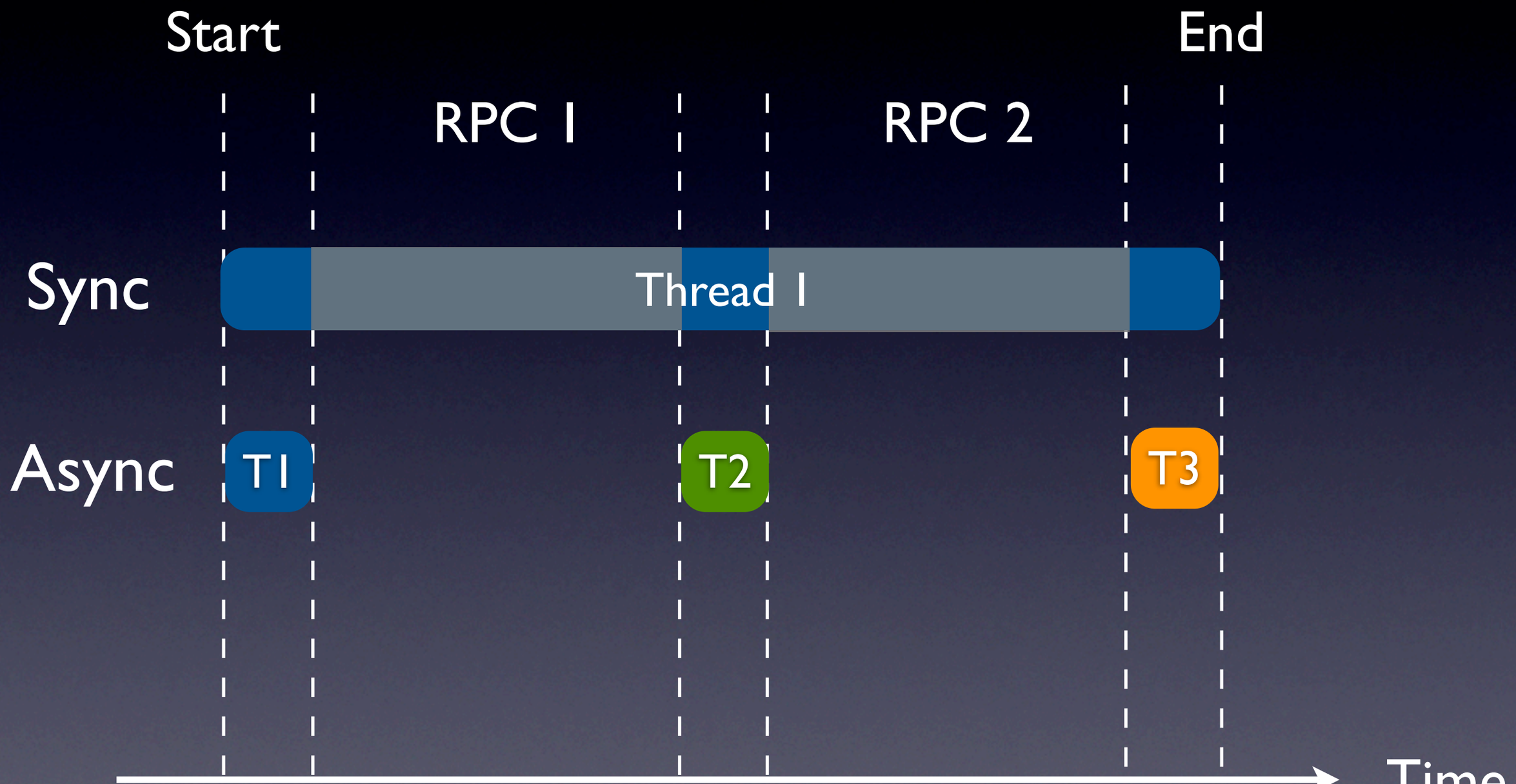
Synchronous Server


```
res handler(req) {
    a = rpc(req.id)
    b = rpc(a)
    return b
}
```

Asynchronous Server


```
handler(req, res) {
    rpc(req.id,
        new Callback() {
          run(a) {
              rpc(a, new Callback() {
                run(b) { res.finish(b) }})
          }
        })
}
```

Lifetime of Request



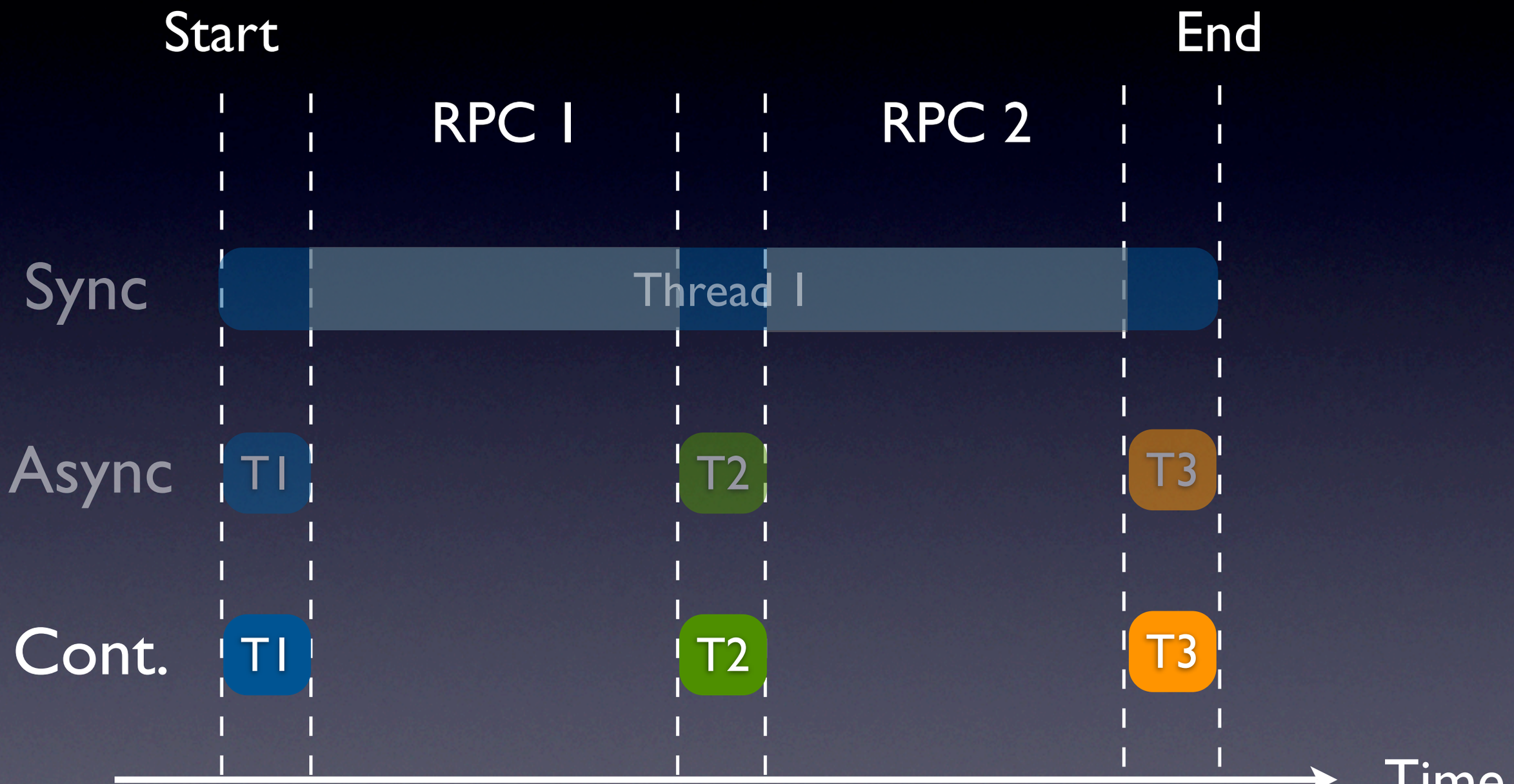
Dilemma

	Scalability	Productivity
Sync		✓
Async	✓	

Continuation Server


```
res handler(req) {
  a = rpc(req.user)
  b = rpc(a)
  return b
}
```

Lifetime of Request

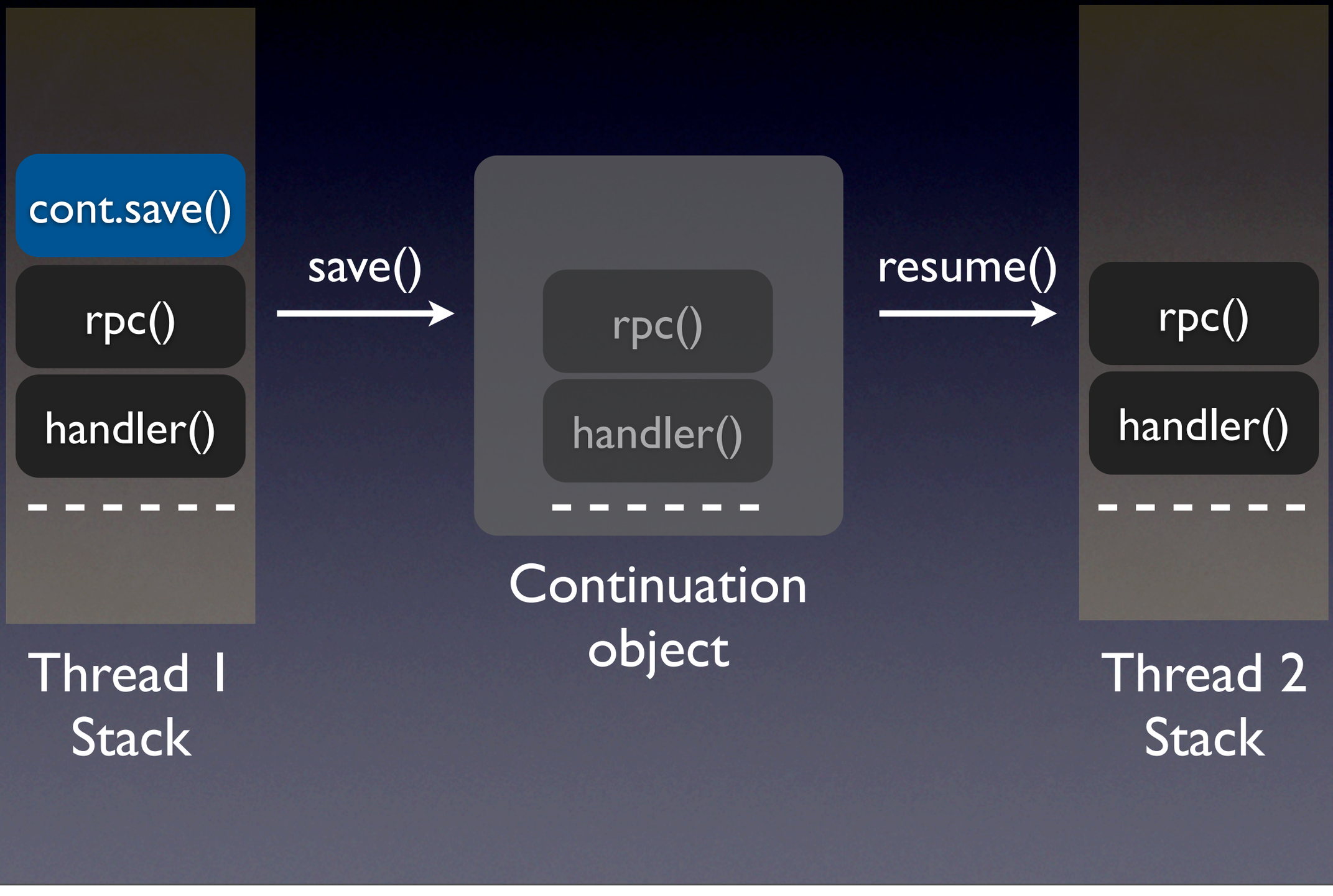


Trick

```
res handler(req) {
  a = rpc(req.user)
  b = rpc(a)
  return b
}
```

```
rpc(p) {
   rpc_async(p,
     new Callback(
       run(r){ cont.resume(r) })
   )
   return cont.save()
}
```

Trick

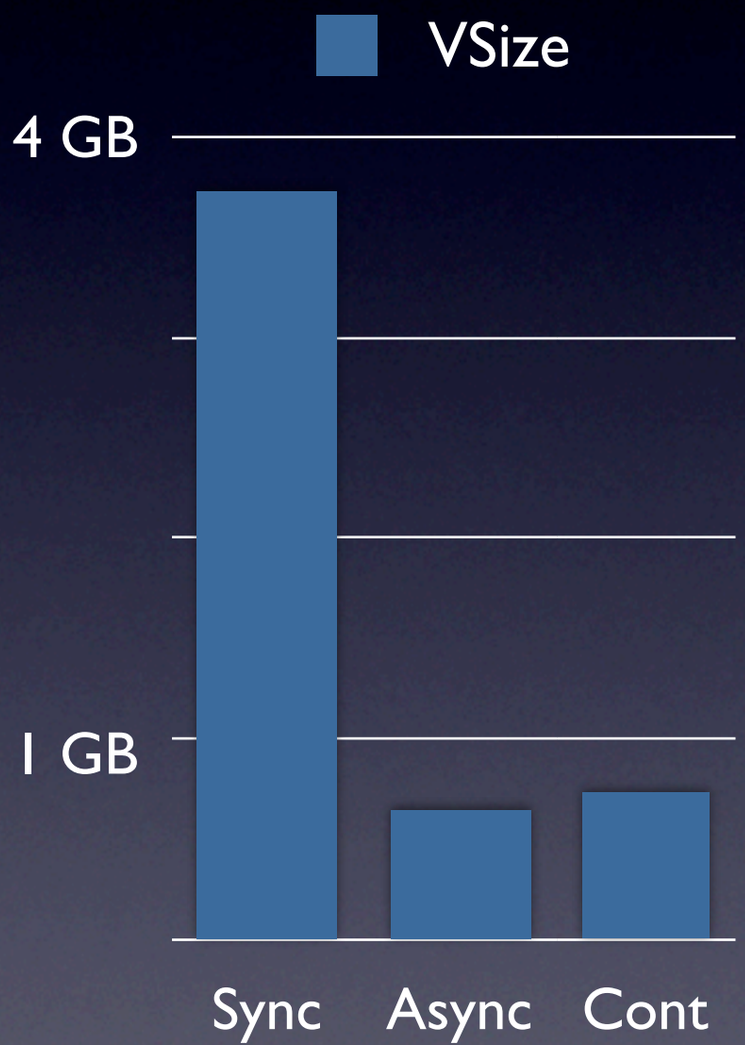

cont.save()
rpc()
handler()
save()
rpc()
handler()
resume()
rpc()
handler()
Thread 1 Stack
Continuation object
Thread 2 Stack

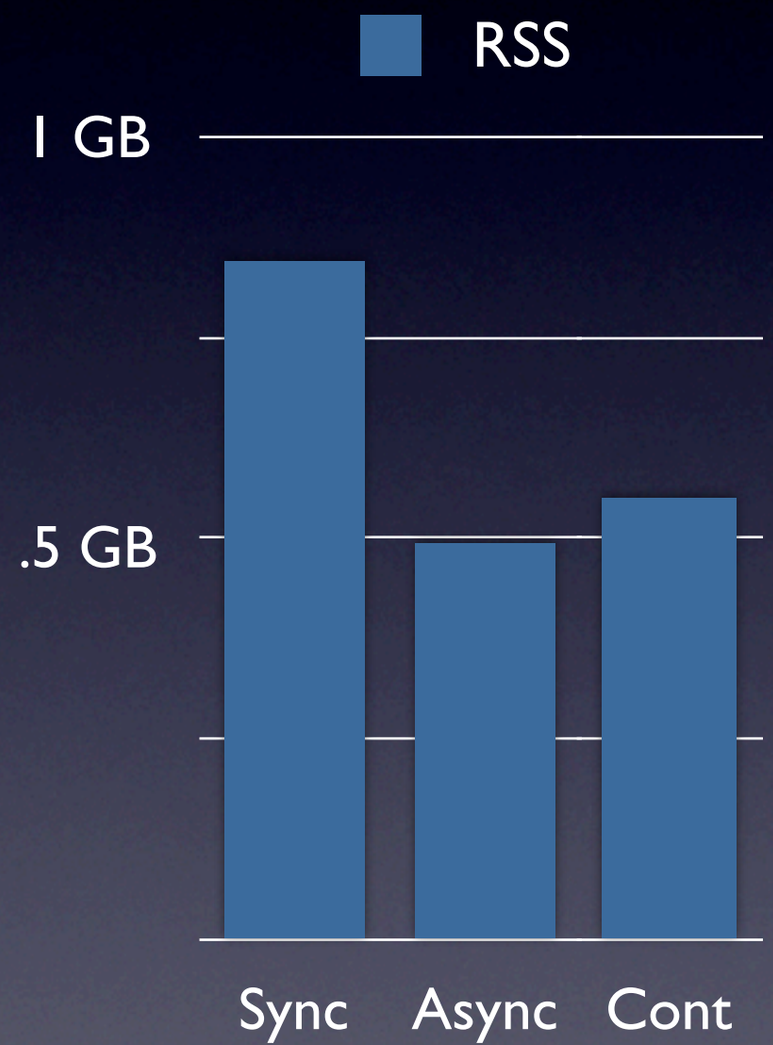
Dilemma

	Scalability	Productivity
Sync		✓
Async	✓	
Cont.	✓	✓

Memory footprint

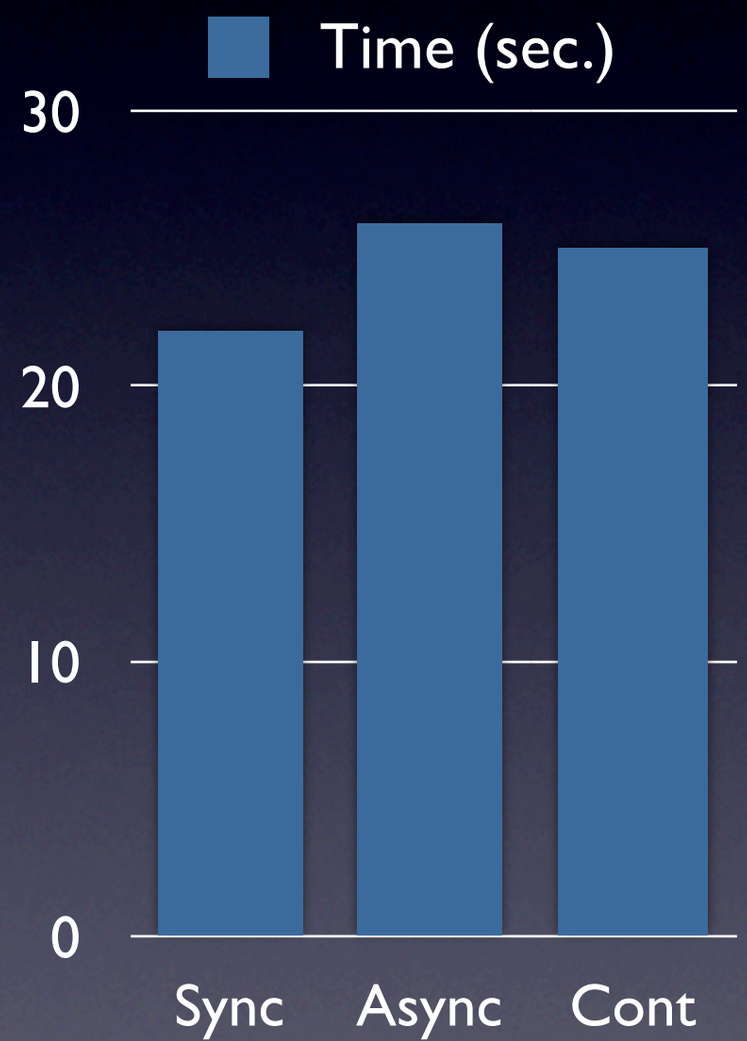
10K requests in flight





Performance

50K Requests Benchmark



Conclusions

Continuation useful in servers

API

```
class Continuation {
  Object save()
  void resume(Object rv)
  static Continuation enter(Runnable scope, Object data1)
  Object getData1()
}
```