

Seattle: Open P2P Computing

Justin Cappos

Computer Science and Engineering

University of Washington



Cloud Computing

Yahoo! CA US IE UK FR NL SG AU NZ

Experian QAS Call us free 1800 213 992

support careers contact us

Home Resources About QAS Products Industry solutions Partners Customers Search... Search

Australia > Resources > Data quality news > Gartner chief: Cloud computing is a very important...

Resources	Data quality news
<ul style="list-style-type: none"> Data quality news Whitepapers & Data Quality Research Reports Webinars Data Management Software Demos Free Software Trials & Data checks Savings Calculator Data Quality Glossary Useful Statistics Events 	<p>Gartner chief: Cloud computing is a very important long-term phenomena</p> <p>- Nov 2 2009, 21:39 PM</p> <p>David Cearley, vice president of research firm Gartner, has said that around 20 per cent of businesses will be using cloud computing technology by 2012.</p> <p>In a speech to the Gartner Symposium ITxpo, Mr Cearley said he expected to see rapid growth in the sector over the next few years, reports iTWire.</p> <p>But he warned that he believed there had been some overhyping of cloud computing and that it would only gradually become one of the dominant IT service techniques.</p> <p>"It's going to be a broad, long-term phenomenon," he explained.</p>

Alter text size T T T

Request more information

Product demonstrations

Request a phone call

Customer case studies

Subscribe to this news...

RSS Get RSS feed

News by category...

General news

Customer Service

Data Quality

Database Management

Contact Us | Create an AWS Account

Support | Your Account

Sign Up For Amazon EC2

Kit

- Windows Azure Tools
- Training Kit

Team Blogs

- Windows Azure
- .NET Services
- SQL Azure

Virtual Private Cloud

- Amazon Virtual Private Cloud

Payments & Billing

- Amazon Flexible Payments Service (Amazon FPS)
- Amazon DevPay

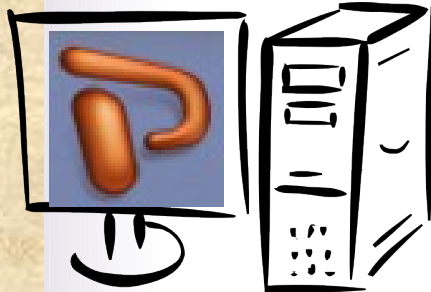
First-time user? register at [Microsoft Connect](#)

Sign in and use Windows Azure

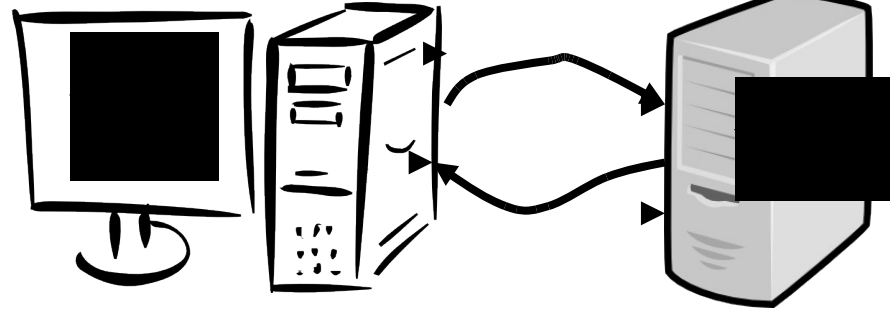
configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate themselves from common failure scenarios.

with G
The Jav

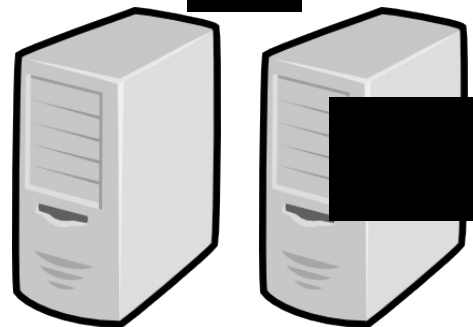
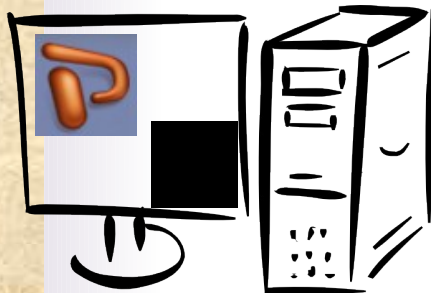
What Cloud Computing Really Is



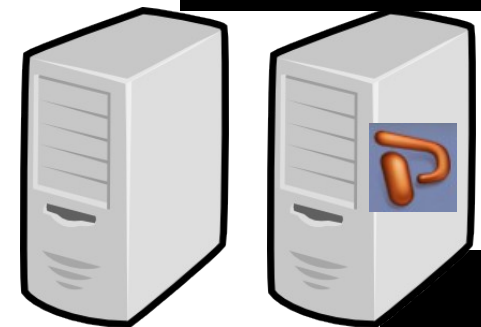
Workstation



Client / Server



Cloud Computing



The Silver Lining

- Computation can scale with demand
- No hardware purchases
- Minimizes IT overhead
- APIs provide
 - Fault tolerance
 - Scalability

The Dark Cloud

- Resources cost money
 - Sponsor w/ advertisements?
- Loss of privacy
- Censorship
- Vendor API lock-in

Cloud computing erodes software freedom

The Peer-to-Peer Alternative

- P2P leverages end user computation
 - Avoid lock-in through open protocols
 - Free
 - Scalability
- P2P challenges
 - Heterogeneity
 - Many apps have short use patterns
 - Fault tolerance
 - Distributed state management

The Peer-to-Peer Alternative

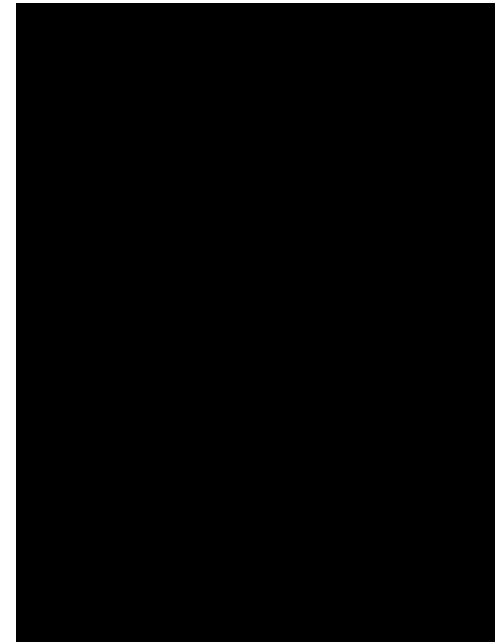
- P2P leverages end user computation
 - Avoid lock-in through open protocols
 - Free
 - Scalability
- P2P challenges
 - Heterogeneity
 - Many apps have short use patterns
 - Fault tolerance
 - Distributed state management

Seattle: Making P2P Easier

- Heterogeneity
 - Provide a portable / virtualized programming abstraction
- Many apps have short use patterns
 - Platform is always-on

What is Seattle?

- P2P hosting platform (it's a potluck!)
 - You contribute some resources
 - You consume what others provide
 - Others consume what you provide
- Contributed resources
 - Remotely accessible
 - Requested through a sharing site
 - Run in a virtual machine (vessel)



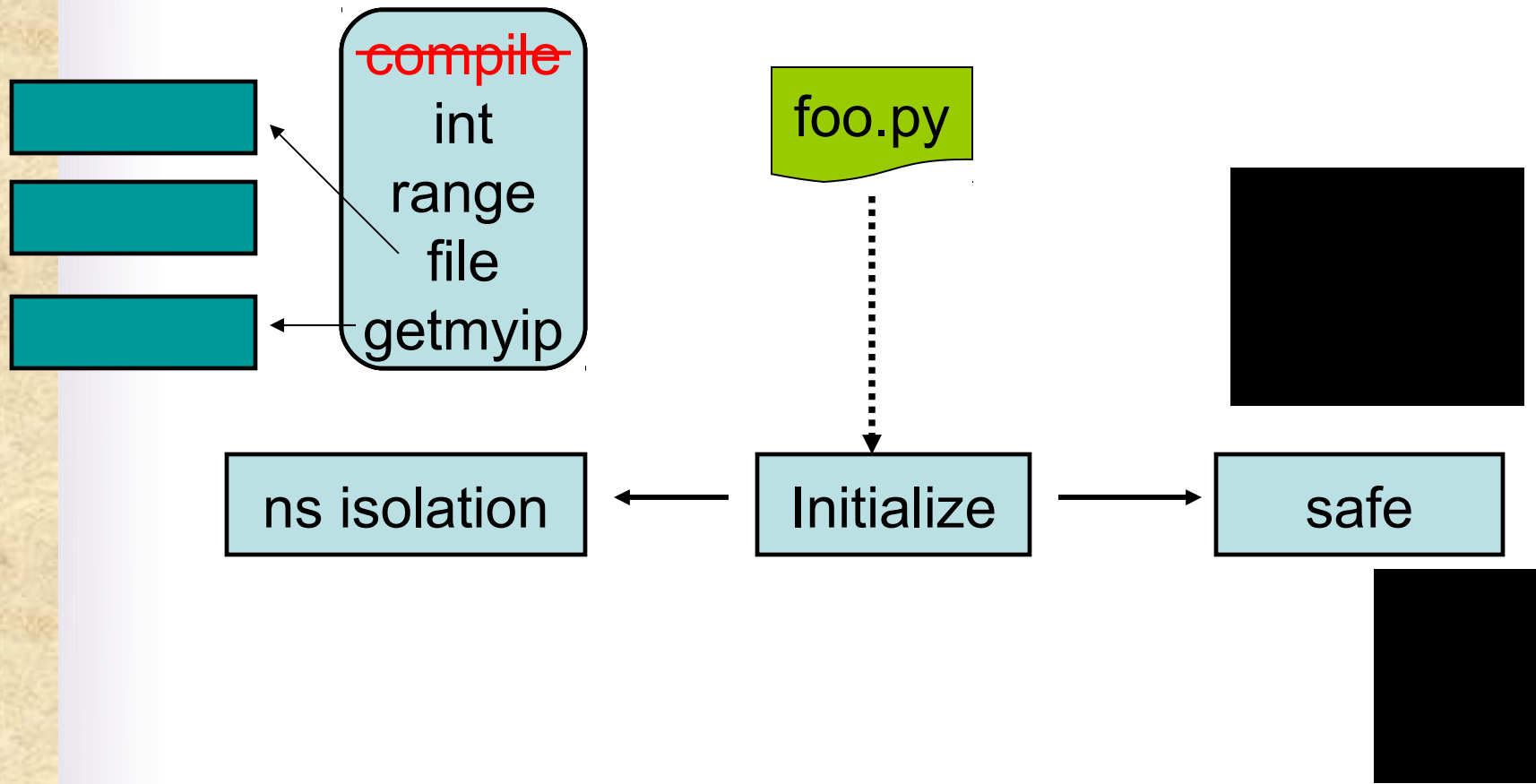
What is a Seattle Vessel?

- Portable / interoperable
 - Mac / Linux / BSD / Windows / Nokia devices, etc.
- Performance isolated
 - CPU, memory, disk, network b/w, etc.
- Safe
 - Isolated directory, can do no harm

Programming Seattle Vessels

- Executes a subset of Python
 - Missing some built-ins, `__slots__`, etc.
 - Replacements for `import`, `eval`, standard libs, etc.
- Implemented in Python
 - Operating system ugliness masked
 - Performance isolation at call time

Executing Python Code Safely



Easy To Code

UDP ping server (4 LOC)

```
def got_message(srcip, srcport, mess, ch):  
    sendmess(srcip, srcport, mess)  
if callfunc == 'initialize':  
    recvmess(getmyip(), 54321, got_message)
```

UDP ping client (6 LOC)

```
def got_reply(srcip, srcport, mess, ch):  
    print 'received:', mess, 'from', srcip, srcport  
if callfunc == 'initialize':  
    recvmess(getmyip(), 43210, got_reply)  
    # send the second arg to the first arg's IP  
    sendmess(callargs[0], 54321, callargs[1], getmyip(), 43210)  
    # exit in one second  
    settimer(1, exitall, ())
```

UDP multicast (<20 LOC), TCP tunneling (<20 LOC), Chord (~300 LOC)

Demonstration

- Registration
- Download Installer
- Acquire resources
 - Use seattlegen website
- Deploy all pairs ping
 - Use shell to locate and control resources
- Automated deployment tools

All Pairs Ping

```
# send a probe message to each neighbor
def probe_neighbors(port):

    for neighborip in mycontext["neighborlist"]:
        mycontext['sendtime'][neighborip] = getruntime()
        sendmess(neighborip, port, 'ping',getmyip(),port)

        sendmess(neighborip, port,'share'+encode_row(getmyip(), mycontext["neighborlist"]
), mycontext['latency'].copy()))
    # sleep a while in case it prevents us from getting a good sample of
    # results e.g. a lot of pings
    sleep(.5)

# Call me again in 10 seconds
while True:
    try:
        settimer(10,probe_neighbors,(port,))
        return
    except Exception, e:
        if "Resource 'events'" in str(e):
            # there are too many events scheduled, I should wait and try again
            sleep(.5)
            continue
        raise
```

Send periodic UDP pings

15 LOC

```
# Handle an incoming message
def got_message(srcip,srcport,mess,ch):
    if mess == 'ping':
        sendmess(srcip,srcport,'share')
    elif mess == 'pong':
        # elapsed time is now - time when I sent the ping
        mycontext['latency'][srcip] = getruntime() - mycontext['sendtime'][srcip]

    elif mess.startswith('share'):
        mycontext['row'][srcip] = mess[len('share'):]
```

Handle incoming UDP pings

7 LOC

```
def encode_row(rowip, neighborlist, latencylist):
    retstring = "<tr><td>"+rowip+"</td>"
    for neighborip in neighborlist:
        retstring = retstring + "<td>"+str(latencylist[neighborip]):4+"</td>"
    else:
        retstring = retstring + "<td>Unknown</td>"

    retstring = retstring + "</tr>"
    return retstring
```

Format latency data into HTML

9 LOC

```
def show_status(srcip,srcport,connobj, ch, mainch):

    webpage = "<html><head><title>Latency Information</title></head><body><h1>Latency information from "+getmyip()+"</h1><table border='1'">

    webpage = webpage + "<tr><td></td><td>"+ ".join(mycontext['neighborlist']
t')+</td></tr>"

    for nodeip in mycontext['neighborlist']:
        if nodeip in mycontext['row']:
            webpage = webpage + mycontext['row'][nodeip]
        else:
            webpage = webpage + "<tr><td>'"+nodeip+"</td><td>No Data Reported</td></tr>\n"

    # now the footer...
    webpage = webpage + '</table></html>'

    # send the header and page
    connobj.send('HTTP/1.0 200 OK\nContent-Length: '+str(len(webpage))+'\nDate: Fri,
31 Dec 1999 23:59:59 GMT\nContent-Type: text/html\n\n'+webpage)

    # and we're done, so let's close this connection...
    connobj.close()
```

Return a webpage

11 LOC

```
if callargs == "initialize":

    # this holds the response information (i.e. when nodes responded)
    mycontext['latency'] = {}
    # this remembers when we sent a probe
    mycontext['sendtime'] = {}
    # this remembers row data from the other nodes
    mycontext['row'] = {}

    # get the nodes to probe
    mycontext['neighborlist'] = []
    for line in file("neighborlist.txt"):
        mycontext['neighborlist'].append(line.strip())

    ip = getmyip()
    if len(callargs) != 1:
        raise Exception, "Must specify the port to use"
    pingport = int(callargs[0])

    # call gotmessage whenever receiving a message
    recvmess(ip,pingport,got_message)

    probe_neighbors(pingport)

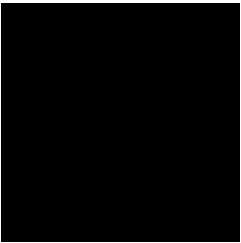
    # we want to register a function to show a status webpage (TCP port)
    pageport = int(callargs[0])
    waitforconn(ip,pageport,show_status)
```

Initialization

15 LOC

Potential Apps

- File backup
- Photo sharing
- Web caching
- Distributed web server
- Email Service
- Web proxy
- ... etc.



Summary

Seattle makes p2p easy

- Safely executes Python
- Portable / interoperable
- Performance isolation
- Open source (MIT license)
- As of Oct 2009, we have resources on over 2000 computers, 100s of cities, 6 continents

P2P keeps free software free!

<https://seattle.cs.washington.edu/>