Getting popular

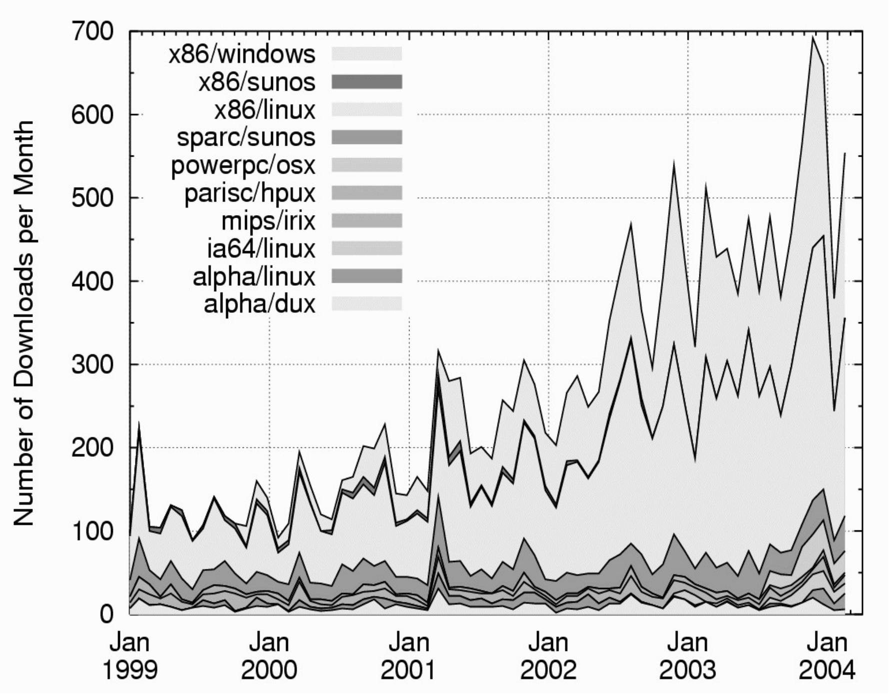


Figure 1: Condor downloads by platform

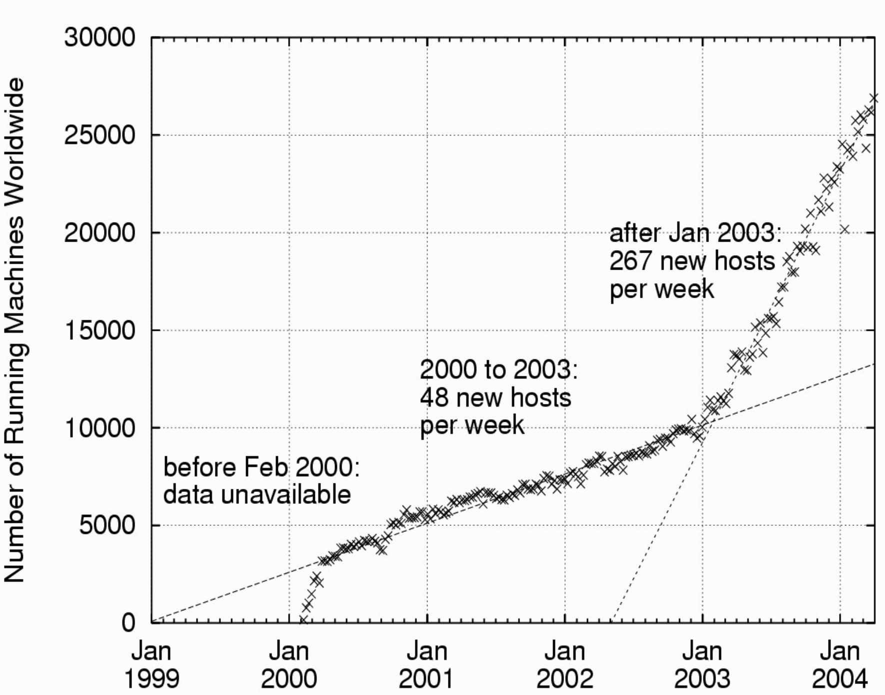
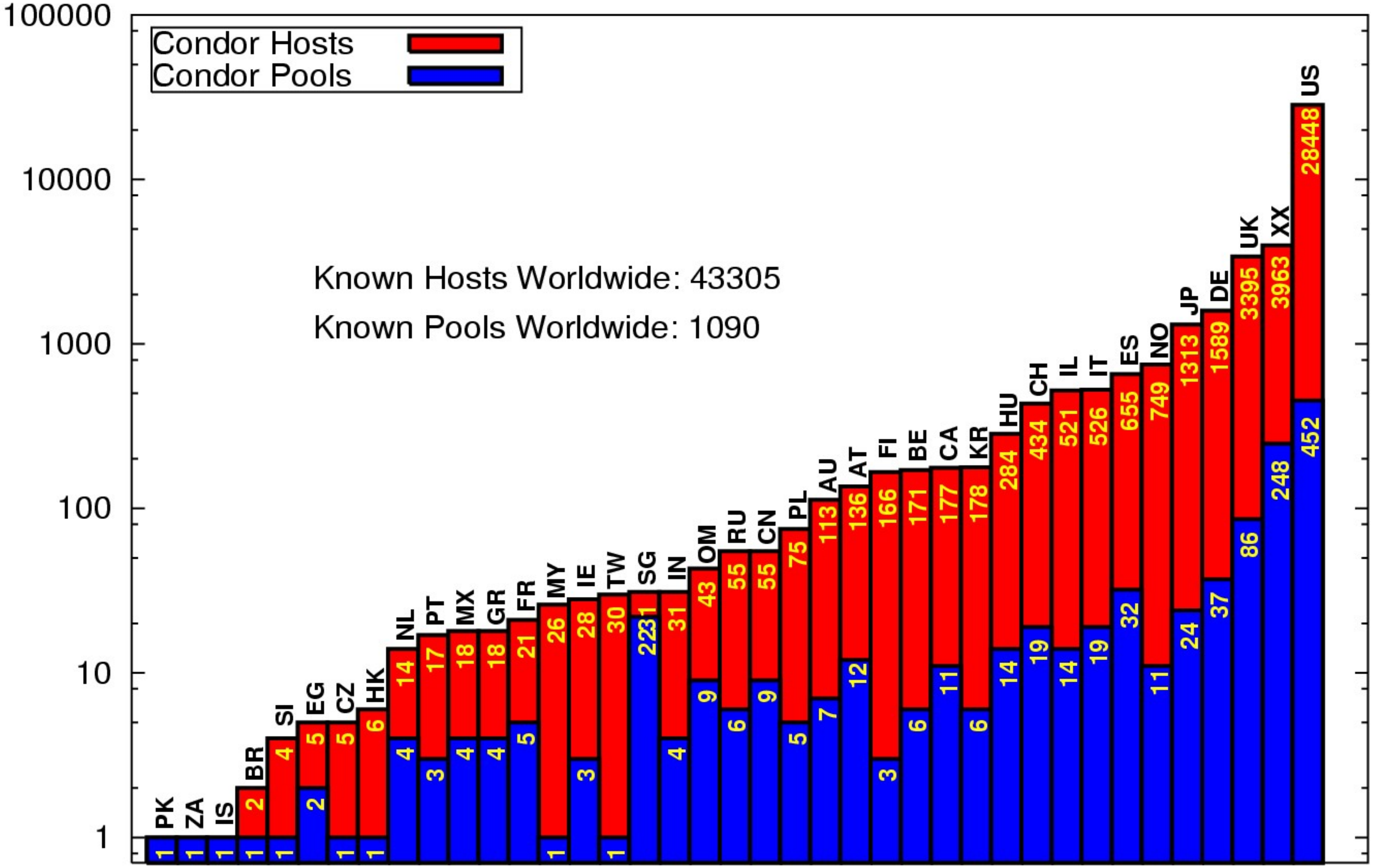


Figure 2: Known # of Condor hosts

http://www.cs.wisc.edu/condor

Known Condor Pools and Hosts by Country
Wed Oct 20 18:38:35 CDT 2004



Known Hosts Worldwide: 43305

Known Pools Worldwide: 1090

Interfacing Applications w/ Condor

- Suppose you have an application which needs a lot of compute cycles
- You want this application to utilize a pool of machines
- How can this be done?

Some Condor APIs

- Command Line tools
  - condor_submit, condor_q, etc
- SOAP
- DRMAA
- Condor GAHP
- MW
- Condor Perl Module
- Ckpt API

Command Line Tools

- Don’t underestimate them
- Your program can create a submit file on disk and simply invoke condor_submit:

```
system(“echo universe=VANILLA > /tmp/condor.sub”);
system(“echo executable=myprog >> /tmp/condor.sub”);
. . .
system(“echo queue >> /tmp/condor.sub”);
system(“condor_submit /tmp/condor.sub”);
```

Command Line Tools

- Your program can create a submit file and give it to condor_submit through stdin:

```
PERL:       fopen(SUBMIT, “|condor_submit”);
            print SUBMIT “universe=VANILLA\n”;
            . . .
C/C++:      int s = popen(“condor_submit”, “r+”);
            write(s, “universe=VANILLA\n”, 17/*len*/);
            . . .
```

Command Line Tools

- Using the +Attribute with condor_submit:

```
universe = VANILLA
executable = /bin/hostname
output = job.out
log = job.log
+webuser = “zmiller”
queue
```

Command Line Tools

- Use -constraint and -format with condor_q:

```
% condor_q -constraint ‘webuser==“zmiller”’
```

```
-- Submitter: bio.cs.wisc.edu : <128.105.147.96:37866> : bio.cs.wisc.edu
 ID       OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
213503.0   zmiller         10/11 06:00   0+00:00:00 I  0   0.0  hostname
```

```
% condor_q -constraint 'webuser=="zmiller"' -format "%i\t"
ClusterId -format "%s\n" Cmd
```

```
213503  /bin/hostname
```

Command Line Tools

- condor_wait will watch a job log file and wait for a certain (or all) jobs to complete:

```
system(“condor_wait job.log“);
```

Command Line Tools

- condor_q and condor_status -xml option
- So it is relatively simple to build on top of Condor's command line tools alone, and can be accessed from many different languages (C, PERL, python, PHP, etc).
- However…

DRMAA

- DRMAA is a GGF standardized job-submission API
- Has C (and now Java) bindings
- Is not Condor-specific -- your app could submit to any job scheduler with minimal changes (probably just linking in a different library)

DRMAA

- Unfortunately, the DRMAA API does not support some very important features, such as:
  - Two-phase commit
  - Fault tolerance
  - Transactions

Condor GAHP

- The Condor GAHP is a relatively low-level protocol based on simple ASCII messages through stdin and stdout
- Supports a rich feature set including two-phase commits, transactions, and optional asynchronous notification of events
- Is available in Condor 6.7.X

GAHP, cont

Example:

```
        R: $GahpVersion: 1.0.0 Nov 26 2001 NCSA\ CoG\ Gahpd $
        S: GRAM_PING 100 vulture.cs.wisc.edu/fork
        R: E
        S: RESULTS
        R: E
        S: COMMANDS
        R: S COMMANDS GRAM_JOB_CANCEL GRAM_JOB_REQUEST GRAM_JOB_SIGNAL
  GRAM_JOB_STATUS GRAM_PING INITIALIZE_FROM_FILE QUIT RESULTS VERSION
        S: VERSION
        R: S $GahpVersion: 1.0.0 Nov 26 2001 NCSA\ CoG\ Gahpd $
        S: INITIALIZE_FROM_FILE /tmp/grid_proxy_554523.txt
        R: S
        S: GRAM_PING 100 vulture.cs.wisc.edu/fork
        R: S
        S: RESULTS
        R: S 0
        S: RESULTS
        R: S 1
        R: 100 0
        S: QUIT
        R: S
```

SOAP

- Simple Object Access Protocol
- Mechanism for doing RPC using XML
  - typically over HTTP
- A World Wide Web Consortium (W3C) standard

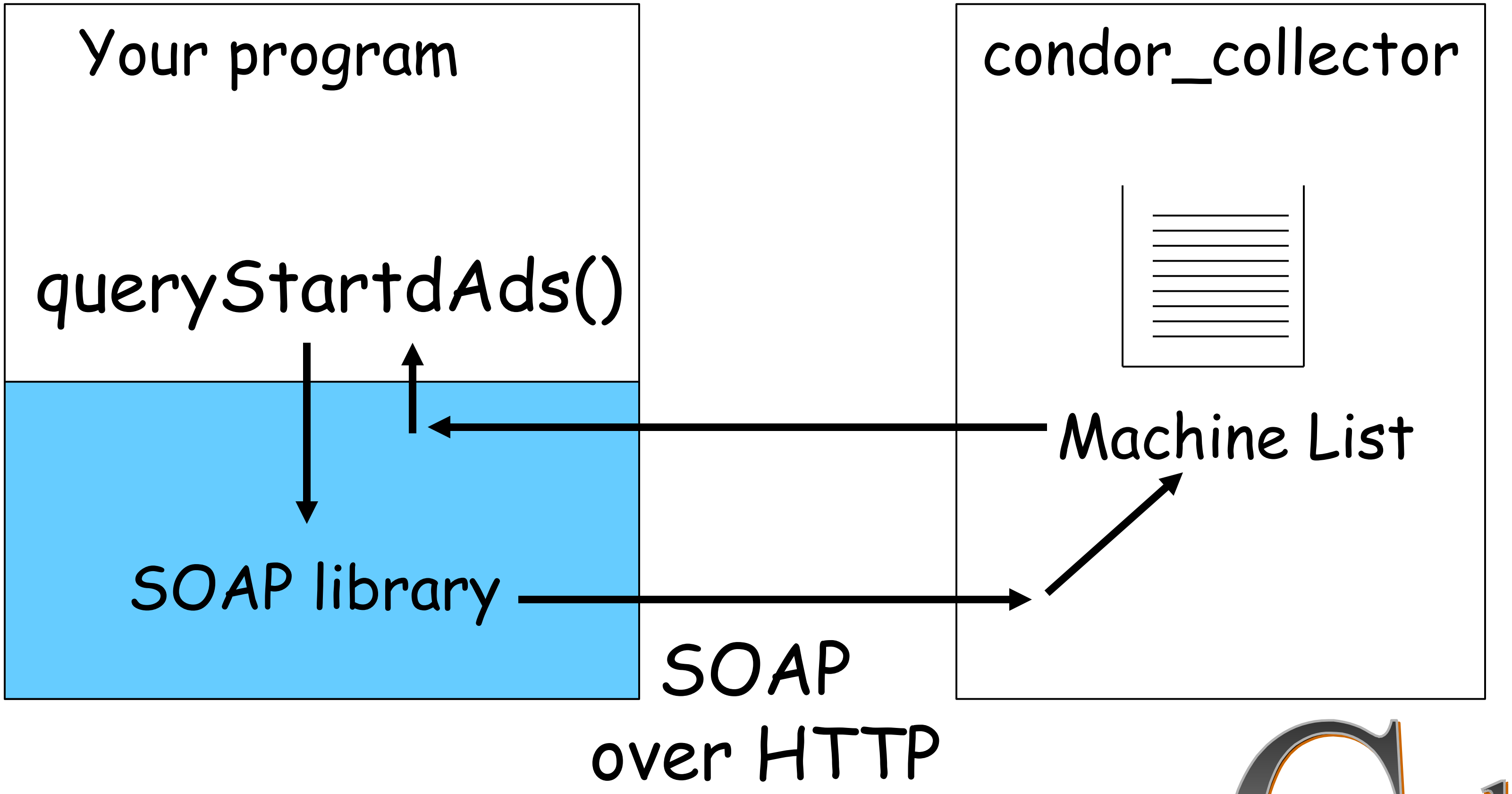
Benefits of a Condor SOAP API

- Condor becomes a service
  - Can be accessed with standard web service tools
- Condor accessible from platforms where its command-line tools are not supported
- Talk to Condor with your favorite language and SOAP toolkit

Condor SOAP API functionality

- Submit jobs
- Retrieve job output
- Remove/hold/release jobs
- Query machine status
- Query job status

Getting machine status via SOAP



Getting machine status via SOAP (in Java with Axis)

```
locator = new CondorCollectorLocator();

collector = locator.getcondorCollector(new
        URL(“http://machine:port”));


ads = collector.queryStartdAds(“Memory>512“);
```

Because we give you WSDL information you don't have to write any of these functions.

Submitting jobs

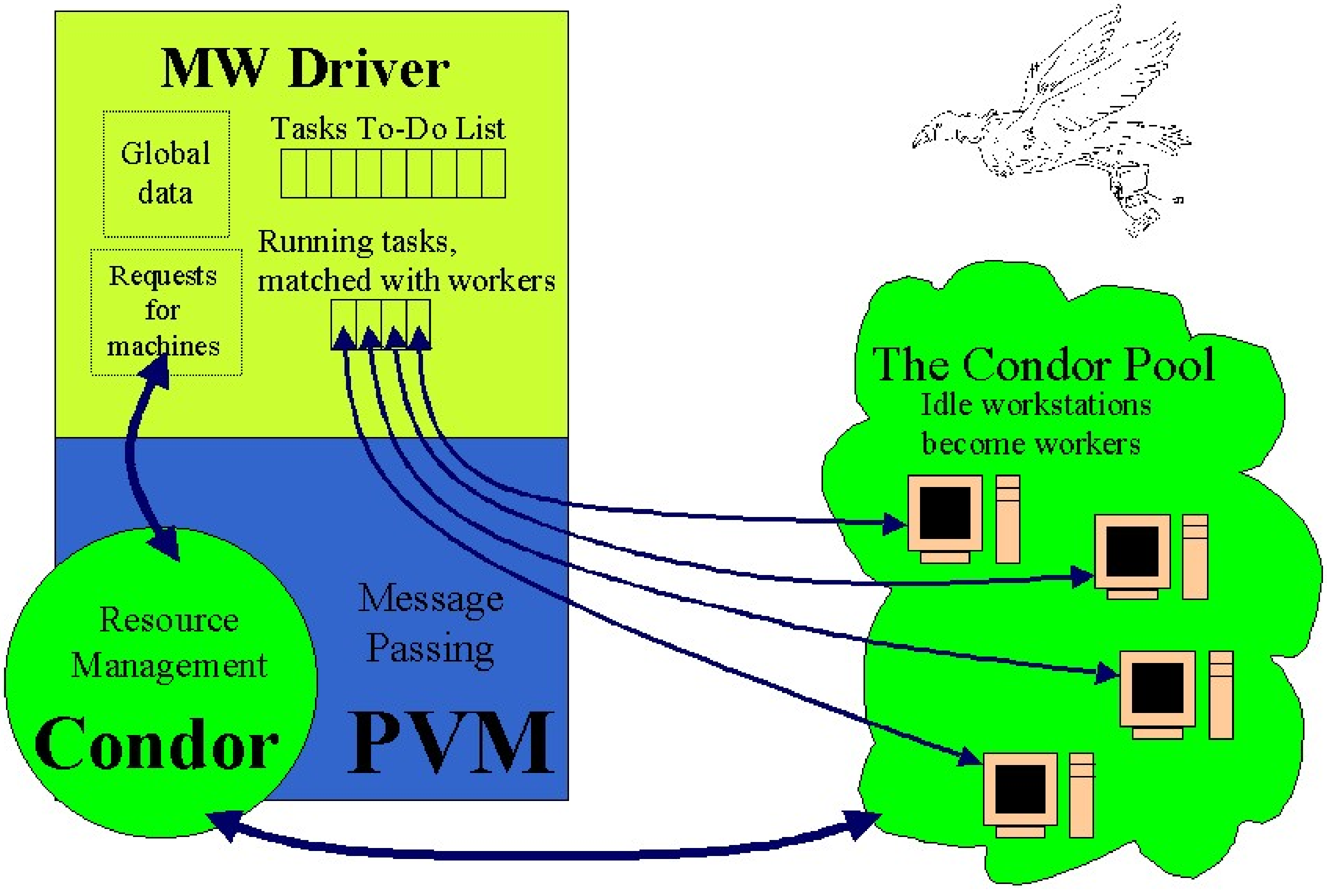
1. Begin transaction
2. Create cluster
3. Create job
4. Send files
5. Describe job

Wash, rinse, repeat

6. Commit transaction
   - Two phase commit for reliability

MW

- MW is a tool for making a master-worker style application that works in the distributed, opportunistic environment of Condor.
- Use either Condor-PVM or MW-File a file-based, remote I/O scheme for message passing.
- Motivation: Writing a parallel application for use in the Condor system can be a lot of work.
  - Workers are not dedicated machines, they can leave the computation at any time.
  - Machines can arrive at any time, too, and they can be suspended and resume computation.
  - Machines can also be of varying architechtures and speeds.
- MW will handle all this variation and uncertainly in the opportunistic environment of Condor.



The Master

The Workers

http://www.cs.wisc.edu/condor

MW and NUG30

quadratic assignment problem

30 facilities, 30 locations

- minimize cost of transferring materials between them

posed in 1968 as challenge, long unsolved

but with a good pruning algorithm & high-throughput computing...

NUG30 Solved on the Grid with Condor + Globus

Resource simultaneously utilized:

- the Origin 2000 (through LSF) at NCSA.
- the Chiba City Linux cluster at Argonne
- the SGI Origin 2000 at Argonne.
- the main Condor pool at Wisconsin (600 processors)
- the Condor pool at Georgia Tech (190 Linux boxes)
- the Condor pool at UNM (40 processors)
- the Condor pool at Columbia (16 processors)
- the Condor pool at Northwestern (12 processors)
- the Condor pool at NCSA (65 processors)
- the Condor pool at INFN (200 processors)

NUG30 - Solved!!!

Sender: goux@dantec.ece.nwu.edu
Subject: Re: Let the festivities begin.

Hi dear Condor Team,

you all have been amazing. NUG30 required **10.9 years** of Condor Time. In just **seven days** !

More stats tomorrow !!! We are off celebrating !

condor rules !

cheers,

JP.

Condor Perl Module

- Perl module to parse the “job log file”
- Recommended instead of polling w/ condor_q
- Call-back event model
- (Note: job log can be written in XML)

“Standalone” Checkpointing

- Can use Condor Project’s checkpoint technology outside of Condor…
  - SIGTSTP = checkpoint and exit
  - SIGUSR2 = periodic checkpoint

```
condor_compile cc myapp.c –o myapp
myapp -_condor_ckpt foo-image.ckpt
…
myapp -_condor_restart foo-image.ckpt
```

Checkpoint Library Interface

- void init image with file name(char *ckpt file name)
- void init image with file descriptor(int fd)
- void ckpt()
- void ckpt and exit()
- void restart()
- void condor ckpt disable()
- void condor ckpt enable()
- int condor warning config(const char *kind,const char *mode)
- extern int condor compress ckpt