MR-Tandem: X!Tandem on Amazon Web Services Elastic Map Reduce (AWS EMR) and Hadoop

1. Overview

There have been other parallelized versions of X!Tandem, but they employ MPI, which many users have found doesn't scale easily (that is, as you add compute nodes the odds of the MPI ring coming up and staying up rapidly decline). By using Hadoop (an implementation of Map Reduce), we have created a parallel X!Tandem which is highly scalable and fault tolerant. And, it's built to work easily in the cloud (AWS, specifically) so you don't have to own your own compute cluster to use it, although that's also supported.

Familiar Operation

MR-Tandem is designed to use the same parameters file you would use with a standard X!Tandem run. It will use the same input parameter files and write its results to the same locations. Input files are transferred to the cloud as needed - if they've been used in a previous run they won't be transferred again.

The MR-Tandem project doesn't provide a fancy front end, it's just a script that you can use where ever you were already calling X!Tandem directly, thus enabling your existing systems to employ the cloud.

Multiple Searches in a Single Run

With MR-Tandem you can replace the xtandem parameters file with a simple text file that's actually a list of xtandem parameter files, and MR-Tandem will execute each of them in turn. This is particularly useful for AWS where time is billed to the nearest hour, and bringing up a cluster for each job leaves a lot of unused time in an hour.

Credits and Source Code

Of course it all begins with X!Tandem, see http://www.thegpm.org/tandem for details.

The code for communicating between the parallel MR-Tandem processes owes much to the X!!Tandem ("X Bang Bang Tandem") project, see http://wiki.thegpm.org/wiki/X!!Tandem for details.

MR-Tandem uses a lightly modified version of X!Tandem, the code for which is maintained as part of the Trans-Proteomic Pipeline, under the Sashimi project (http://sourceforge.net/projects/sashimi/). But you don't have to think about that, prebuilt MR-Tandem binaries for most linux versions are available on Amazon S3 in the "insilicos-public" bucket and the launcher script grabs them for you.

If you're interested, though, the modified X!Tandem source code is available at
https://sashimi.svn.sourceforge.net/svnroot/sashimi/trunk/trans_proteomic_pipeline/extern/xtandem
You'll need to build it within the context of TPP:
cd /sashimi/trunk/trans_proteomic_pipeline/installer_linux ; ./publish_tandem_to_s3.sh
But again, normally you automatically use the binaries that Insilicos publicly hosts on S3.

This work was funded by NIH grant R43HG006091.

MR-Tandem is part of the **Insilicos Cloud Army** project, see http://sourceforge.net/projects/ica/ for details.

2. Getting Started

Installer for Windows Users

Refer to the accompanying Quick Start Guide for getting a fast start on Windows.

Manual Installation for Linux users

Whether you use AWS or your own cluster, there a few local setup steps required:

Make sure you have Python 2.5 or higher installed. See http://www.python.org/download/releases/2.7.2/ for details. Python 3.x hasn't been tested, so stay with 2.x.

Also you'll need Boto 2.0rc1 or newer (boto is a python package for talking to Amazon Web Services, but you'll need it even if you're using your own Hadoop cluster). Visit http://pypi.python.org/pypi/boto for latest versions. Run "setup.py install" from the boto directory once you have it on your local machine.

And you'll need the simpleJSON package: get http://pypi.python.org/packages/source/s/simplejson/simplejson-2.1.1.tar.gz and run "setup.py install" from the simplejson-2.1.1 directory.

For your convenience on linux, there's a script provided for the Boto and simpleJSON steps, run "install_prerequisites.sh".

3. Run With Amazon Web Services Elastic MapReduce (AWS EMR)

You'll need an account on AWS, and you'll need to be signed up for EC2, S3, and EMR. Visit http://aws.amazon.com/ for all the details on that.

To run a job, it's

```
mr-tandem.py <your-aws-info-in-a-json-file> <xtandem-params-file> [<additional parameters as json>]
```

The mr-tandem.py script handles getting your data out to S3 and making the run happen on AWS. It gets the results back to your local drive as well. It grabs a prebuilt xtandem binary from a public S3 bucket hosted by Insilicos. Files which are already on S3 from previous runs (FASTA files, for example) won't be uploaded again if the local copy hasn't been changed.

The final results will appear on your local computer with all the internal file references consistent with the local filenames used in the xtandem params file, even though the files were actually on the cluster.

Your AWS info file should look something like this:

```
{
  # this is a comment.
  "aws_access_key_id": "000000DEADBEEF00000000",
  "aws_secret_access_key": "000000FEEDDEADBEEF000000000000000",
  "RSAKeyName": "my-keypair",
  "RSAKeyFileName": "C:/myawsstuff/my-keypair.pem",
  "s3bucketID": "mys3bucket"
}
```

4. Setting the Cluster Size and Other Parameters You May Wish to Adjust

You can add these parameters to your AWS info file, or at the commandline, or in a secondary config file. For example, to specify a cluster with 8 nodes, you could add this line to your AWS info file

```
"numberOfClientNodes": "8",
```

or you could add this to the commandline

```
{"numberOfClientNodes":"8"}
```

note: you may need to type this as `{\"numberOfClientNodes\":\"8\"}`

Or you could create a file "myfile.json" containing the line

```
{"numberOfClientNodes":"8"}
```

And add "`myfile.json`" to the commandline.

These are the most commonly used parameters:

- `numberOfClientNodes`: Number of client nodes you want to spin up in the cloud. Default: 4.

- `numberOfMappers`: An alternate way to specify cluster size, this sets the number of processes used. The number of client nodes will depend on the type of client node selected – MR-Tandem tries to run one process per core.

- `spotBid`: Allows you to bid for idle AWS compute resources that would otherwise go unused. This is less expensive, but there's a possibility of being interrupted before you're done (you aren't charged for partial hours when that happens, though). You can express your bid as a dollar amount per node per hour (e.g. "0.52") or as a percentage of the "demand instance" price (e.g. "75%%" - note the doubled % sign is needed for some systems command line processing). You generally pay less than you bid, so going high doesn't hurt. A job with a spotBid may take longer to start, since it has to wait for AWS to decide whether to accept the bid or not.
  See http://aws.amazon.com/ec2/spot-instances/ for full information on how spot instances work.

- `verbose`: Set to `True` to get more voluminous logging. Default: `False`.

- `debug`: Set to `True` to get even more detailed logging. Default: `False`.

- `ec2_head_instance_type`: Specifies the instance type of your head node. Recognized values include `t1.micro`, `m1.small`, `m1.large`, `m1.xlarge`, `m2.large`, `m2.2xlarge`, `m2.4xlarge`, `c1.medium`, `c1.xlarge`, `cc1.4xlarge` and `cg1.4xlarge`. If not specified, defaults to `m1.large`. See http://aws.amazon.com/ec2/instance-types/ for more information on EC2 instance types.

- `ec2_client_instance_type`: Similar to `ec2_head_instance_type`, but for client nodes. If not specified, defaults to the current value of `ec2_head_instance_type`. Note that not all combinations of head and client instance types will work – the architectures must match. So you could combine an `m1.small` (32-bit) head node with `c1.medium` (32-bit, high CPU) client nodes, but not with `m1.large` (64-bit) client nodes.

- `aws_region` and `aws_placement`: It's usually best to leave these unspecified so that AWS can choose the least busy availability zone placement for you. If you do wish to specify, acceptable `aws_placement` values are single lower case letters – for example, if you set `aws_region` to "us-east-1" and you set `aws_placement` to "b", your job would run in us-east-1b.

5. Launching Multiple Jobs in a Single Cluster Instance

EC2 charges by the hour, so a cluster instance that runs for 15 minutes costs the same as one that runs for 60 minutes. Thus 3 jobs that take 15 minutes each would complete in under an hour, but your cost would be for 3 hours of cluster time if you run them in separate cluster instances. Happily, it's easy to run them all in a single instance.

For example, you could replace the creation of these three cluster instances:

```
mr-tandem.py my_general_config.json job1_xtandem.params
mr-tandem.py my_general_config.json job2_xtandem.params
mr-tandem.py my_general_config.json job3_xtandem.params
```

with this single cluster instance that runs all three jobs before exit:

```
mr-tandem.py my_general_config.json joblist.txt
```

where joblist.txt is a simple text file containing these lines:

```
job1_xtandem.params
job2_xtandem.params
job3_xtandem.params
```

which will cost 1/3 as much.

6. Run With a Private Hadoop Cluster

A local copy of Hadoop is needed to move files on and off the cluster.

Note for Windows users:
Hadoop is very linux-centric and as of this writing there is no official support for it on windows. You'll either have to install Cygwin to get a linux-like shell, or install the free VMWare Player and run a virtual linux machine on your windows box. We prefer the VMWare route but either should work. Details of getting those working are beyond the scope of this document, but google will show you lots of references.

Install Java.
You'll want to use Sun Java SDK, visit: http://java.sun.com/javase/downloads/widget/jdk6.jsp
Install Hadoop Locally
Important!!!! You should install a version of Hadoop that EXACTLY MATCHES the one running on your cluster or the file transfers probably won't work.
See http://wiki.apache.org/hadoop/GettingStartedWithHadoop for details on installing Hadoop.
In my case I downloaded
http://archive.cloudera.com/cdh/testing/hadoop-0.20.1+152.tar.gz
and unpacked it in /usr/local/hadoop-0.20.1+152.
You'll need to update the core-site.xml and mapred-site.xml Hadoop client config files as directed by your cluster admin.

Establish a proxy link to the cluster with this command:

ssh -D <proxy_port> -n -N <your_username>@<your_cluster_URL>

Your admin will tell you what values to use for:

proxy_port
your_username
your_cluster_URL

so it will be something like

```
ssh -D 6789 -n -N joeuser@gw.joescluster.com
```

Open up a new shell and ssh to your cluster, leaving the session open (this ssh connection will pass files back and forth to the cluster).

Now test your connection by running

```
hadoop fs -ls /
```

You'll need to add a few more entries to your main config file:

```
  "hadoop_home": "<your hadoop install directory>", # directory below which
"bin/hadoop" can be found
  "tandem_url": "<something like http://insilicos-
public.s3.amazonaws.com/centos/x86_64/tandem>",  # where to get the tandem
executable (can be a local file if you build your own)
 "hadoop_dir": "<your hdfs dir on cluster>",  # your HDFS directory on the
cluster
 "numberOfMappers": "<number of mappers you want to ask the cluster for>"
```

and optionally you can include these to help when you need reminding to start the proxy:

```
  "hadoop_gateway": "<your_hadoop_gateway>", # your cluster gateway
  "hadoop_user": "<your_hadoop_username>",  # your username on the cluster
```

For Cygwin you'll probably need to repeat the installation steps for simplejson and boto in a Cygwin shell.

7. Run With MPI on AWS EC2

The MR-Tandem binary also implements MPI-based X!!Tandem. MPI setup on EC2 is done by leveraging off another Insilicos Cloud Army project, "Ensemble Cloud Army" (ECA) which uses MPI for ensemble machine learning techniques using R. For our purposes we ignore the "R" aspect and just reuse the MPI infrastructure ECA provides.

If you've followed the installation steps for MR-Tandem running on AWS EMR then you should be fine to run on EC2 with MPI. Just add {\"runBangBang\":\"true\"} to your commandline, or add it to your config file. The Windows installer provides a batch file “bb-tandem.bat” which handles this for you.

Information and code for ECA can be found in the "ensemble" directory which is sibling to this directory in the Insilicos Cloud Army code tree.

8. Troubleshooting

If you have trouble getting a job to run, try running it locally first. The xtandem parameters files used in MR_Tandem work the same way with a local copy of xtandem, so if it's not working locally then you know why it's not working on your cluster. You can get a binary for local use from http://insilicos-public.s3.amazonaws.com - just drill down from there for your compute platform's binary. Windows users should look for the "mingw" directory.