launcher_creator.py

Create SGE or SLURM launchers for Stampede and Lonestar

launcher_creator.py is a convenience utility that automates creating a launcher for TACC clusters based on parameters (and commands) you give it. It outputs a either a SLURM or SGE file that can be submitted to Stampede or Lonestar using `sbatch` or `qsub`. (It detects which machine it is being executed on, so the same code should work on either cluster.) It can run both Bash commands on a single node and/or a parametric list of commands to be dispatched to multiple nodes.

How To Use It

At the command line:

```
$ launcher_creator.py <options listed below>
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n name</td>
<td>The name of the job.</td>
</tr>
<tr>
<td>-a, -A allocation</td>
<td>The allocation you want to charge the run to.</td>
</tr>
<tr>
<td>-q queue</td>
<td>The queue to submit to, like 'normal' or 'largemem', etc.</td>
</tr>
<tr>
<td>-w wayness</td>
<td>Optional. The number of jobs in a job list you want to give to each node. (Default is 12 for Lonestar, 16 for Stampede.)</td>
</tr>
<tr>
<td>-N number of nodes</td>
<td>Optional. Specifies a certain number of nodes to use. You probably don't need this option, as the launcher calculates how many nodes you need based on the job list (or Bash command string) you submit. It sometimes comes in handy when writing pipelines.</td>
</tr>
<tr>
<td>-t time</td>
<td>Time allotment for job, format must be hh:mm:ss.</td>
</tr>
<tr>
<td>-e email</td>
<td>Optional. Your email address if you want to receive an email from Lonestar when your job starts and ends.</td>
</tr>
<tr>
<td>-l launcher</td>
<td>Optional. Filename of the launcher. (Default is <code>&lt;name&gt;.sge</code>)</td>
</tr>
<tr>
<td>-m modules</td>
<td>Optional. String of module management commands. <code>module load launcher</code> is always in the launcher, so there's no need to include that.</td>
</tr>
<tr>
<td>-b Bash commands</td>
<td>Optional. String of Bash commands to execute.</td>
</tr>
<tr>
<td>-j Command list</td>
<td>Optional. Filename of list of commands to be distributed to nodes.</td>
</tr>
<tr>
<td>-s stdout</td>
<td>Optional. Setting this flag outputs the name of the launcher to stdout.</td>
</tr>
</tbody>
</table>

Note that while the "-b" and "-j" options are both optional, using neither would pass no work to Lonestar, so you want to use at least one of them!

Manage Modules

Use the `-m` option to load and swap modules for the script. Use the exact same syntax to load or swap module as you would at the command line.

**Example:** `-m "module load luatools"

To run several module commands in the launcher, just separate the commands with a semicolon, as you would in the shell.

**Example:** `-m "module swap intel gcc; module load bedtools"

Or you can use the shortcut `ml` command to perform several module actions in one command.

**Example:** `-m "ml -intel gcc bedtools"`
Run multi-line shell commands

The \texttt{-b} option lets you insert shell commands into the launcher script. For a short number of commands, separating them with a semicolon may be convenient. For longer commands you may want to write the commands on several lines. To insert a multi-line bash script into a \texttt{launcher_creator.py} command, you can use a \texttt{heredoc} like so:

\begin{verbatim}
Use a heredoc to assign multiple lines to a variable, then use this variable as input to the \texttt{-b} option.
jelly_bash=$(cat <<-JELLY
ulimit -s unlimited
Trinity.pl --seqType $seqtype --JM 900G $input_data --output $trinity_dir
--CPU $cpu_per_node --no_run_chrysalis
JELLY
)

launcher_creator.py -n jelly -t 6:00:00 -b "$jelly_bash"
\end{verbatim}

Create and submit a launcher in one line using \texttt{-s}

The \texttt{-s} option outputs the name of the launcher to STDOUT. With command substitution you can immediately submit the new launcher file.

\begin{verbatim}
Immediately submit a launcher file on Stampede

sbatch $(launcher_creator.py -n variant_calling -t 3:00:00 -q normal -b
"freebayes -f mapping/bwa_TAIR10/TAIR10_all.fasta alignments.sorted.bam>
alignments.vcf" -w 1 -s)
\end{verbatim}

Examples

Distributing Shrimp to four nodes

\begin{itemize}
  \item Make a list of the Shrimp commands to run. You can do this in a text editor, algorithmically with a Bash loop, whatever. Let's pretend this text is saved as \texttt{mapping_commands.list}
\end{itemize}

\begin{verbatim}
List of Shrimp commands

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>gmapper-1s -N 6 --qv-offset 33 -p opp-in --fastq -1 Sample_R1-00.fq -2</td>
</tr>
<tr>
<td>Sample_R2-00.fq reference.fasta &gt; Sample_mapped-00.sam</td>
</tr>
<tr>
<td>gmapper-1s -N 6 --qv-offset 33 -p opp-in --fastq -1 Sample_R1-01.fq -2</td>
</tr>
<tr>
<td>Sample_R2-01.fq reference.fasta &gt; Sample_mapped-01.sam</td>
</tr>
<tr>
<td>gmapper-1s -N 6 --qv-offset 33 -p opp-in --fastq -1 Sample_R1-02.fq -2</td>
</tr>
<tr>
<td>Sample_R2-02.fq reference.fasta &gt; Sample_mapped-02.sam</td>
</tr>
<tr>
<td>gmapper-1s -N 6 --qv-offset 33 -p opp-in --fastq -1 Sample_R1-03.fq -2</td>
</tr>
<tr>
<td>Sample_R2-03.fq reference.fasta &gt; Sample_mapped-03.sam</td>
</tr>
</tbody>
</table>
\end{verbatim}
• **Generate a launcher using** `launcher_creator.py`.

```bash
Using launcher_creator.py

```launcher_creator.py``` -q normal -a MyAllocation -n MapSample -j mapping_commands.list -w 2 -t 6:00:00 -l map_sample.sge -m "module load shrimp"

• **Submit the newly-created launcher.**

```bash
qsub the new launcher on Lonestar

qsub map_sample.sge

```

```bash
sbatch the new launcher on Stampede

sbatch map_sample.slurm

```