python-csp
CSP as a DSL for Python and Jython

Sarah Mount, University of Wolverhampton

<table>
<thead>
<tr>
<th>w:</th>
<th><a href="http://www.snim2.org">http://www.snim2.org</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>e:</td>
<td><a href="mailto:s.mount@wlv.ac.uk">s.mount@wlv.ac.uk</a></td>
</tr>
<tr>
<td>t:</td>
<td>@snim2</td>
</tr>
</tbody>
</table>
Contents...

- Why we did this
- Advantages / disadvantages of Python for highly concurrent or process oriented work
- General theme of python-csp
- Syntax / semantics / examples
- Future directions
The story of this work ...
Tmote Invent platform from MoteIV (now Sentilla)
TinyOS code to gather raw data from Tmote Invents
module HL2ControllerM
{
    provides interface StdControl;
    uses { ... interface ADC as AccelX; ... }
}
implementation {
    task void getAccelXData() {
        call AccelX.getData();
    }
    async event result_t AccelX.dataReady(uint16_t data) {
        atomic am->accelX[nextX++] = data;
        post getAccelYData();
        return SUCCESS;
    }
}
... but what about the application layer?
Soil science and agronomy
\[ \psi_m = \frac{R \cdot T \ln(\frac{\%RH}{100})}{V_w} \]

\[ R = 8.314 \text{ J mol}^{-1} \text{ K}^{-1} \]

\[ V_w = 18 \text{ mL/mol} \]

%RH to Soil Matric Pressure
Then I made SenSor and Dan Goldmsmith made SensorPlus
Laboratory hardware running Dingo
... so we wrote a book about it all ...
Why python-csp

- Keep all the increased productivity and fun of Python
- Add scalable, mobile concurrency
- Profit.
## Commstime results

<table>
<thead>
<tr>
<th></th>
<th>Mean (micro s)</th>
<th>s.d.</th>
</tr>
</thead>
<tbody>
<tr>
<td>JCSP (Java threads)</td>
<td>23.8</td>
<td>4.29</td>
</tr>
<tr>
<td>PyCSP (Processes)</td>
<td>394.97</td>
<td>75.82</td>
</tr>
<tr>
<td>PyCSP (Threads)</td>
<td>292.2</td>
<td>47.21</td>
</tr>
<tr>
<td>PyCSP (Greenlets)</td>
<td>24.41</td>
<td>0.36</td>
</tr>
<tr>
<td>python-csp (Processes)</td>
<td>116.75</td>
<td>35.53</td>
</tr>
<tr>
<td>python-csp (Threads)</td>
<td>225.77</td>
<td>17.51</td>
</tr>
<tr>
<td>jython-csp (Java threads)</td>
<td>157.8</td>
<td>30.78</td>
</tr>
</tbody>
</table>
Python oddities

- The Beazley effect
  - A multi-threaded algorithm can be slower than a single-threaded algorithm
  - GIL preempts every $X$ OPCODES
- The state of Python's low-level threading libraries
  - Implement POSIX threads
  - Locking facilities (condition variables, locks, mutexes, semaphores) usually implemented in natively in Python, not provided by the OS
Morals of this story...

- Not every language has nice, high-level concurrency features
- It is still worth porting CSP etc. to your favourite language
  - If you don't like Python, try Actionscript ;-) 
- The JVM is not the answer to every ill
- Sometimes waiting is a good idea ... 
  - Google will finish Unladen Swallow 
  - Jython will get faster (but will it get jythonc back?!)


Future directions

• Mobility
• Performance issues
  o Can we do better?
    ▪ Coroutines, protothreads, ...
    ▪ Unladen Swallow (LLVM -> ???)
• Using the underlying thread / process libraries
  o Brings an overhead
  o Doesn't directly implement POSIX anything
  o May prove useful to replace
• Pythonic issues
  o Get high level concurrency into the standard library ;-) 
• Pervasive computing -- bigraphs?