SWIFT & SWIFT/T

Presented By:
Adam Ordway, Ben Johnson, and Brandon Sloan

What is Swift?

- Swift is not Apple's Swift.
- Swift is a fast and simple scripting language with the intent of **parallel** programming.
- Swift is concurrent language with syntax similar to C and built with Java.
- Swift contains features of other languages like arrays, for/foreach loops, if/else statements, and functions.

Benefits of Swift

- If tasks are able to run concurrently, i.e. are not dependent on each other, they will.
- One script can run on multiple cores, nodes, grids, etc.
- Incremental viewable updates (see figure below).
- Able to be run on any machine you have available.

```
y  = f(x);
z1 = g(y, 1);
z2 = g(y, 2);
```

```
$ swift p3.swift -nsim=100 -steps=1
Swift 0.96
RunID: run002
Progress: Thu, 22 Jan 2015 16:29:45-0600
Progress: Thu, 22 Jan 2015 16:29:46-0600  Selecting site:80  Active:20
Progress: Thu, 22 Jan 2015 16:30:07-0600  Selecting site:60  Active:20  Finished
successfully:20
Progress: Thu, 22 Jan 2015 16:30:28-0600  Selecting site:40  Active:20  Finished
successfully:40
Progress: Thu, 22 Jan 2015 16:30:49-0600  Selecting site:20  Active:20  Finished
successfully:60
Progress: Thu, 22 Jan 2015 16:31:10-0600  Active:20  Finished successfully:80
Final status:Thu, 22 Jan 2015 16:31:31-0600  Finished successfully:101
```

Example

```cpp
// Example wasteful code
// The point of this is to waste time
//  in order to simulate large data
//  to maximize the effects of parallelization

#include <iostream>
using namespace std;

// makes wasteful calculations
int multdiv(int i) {
    i *= 10000;
    i /= 3;
    i *= 3000;
    i /= 42;
    return i;
}


int main() {
    int num;

    // takes numbers on stdin
    // and runs the wasteful calculations
    // on each one
    while (cin >> num) {
        cout << multdiv(num) << endl;
    }

    return 0;
}

```

Linear code: Bash script

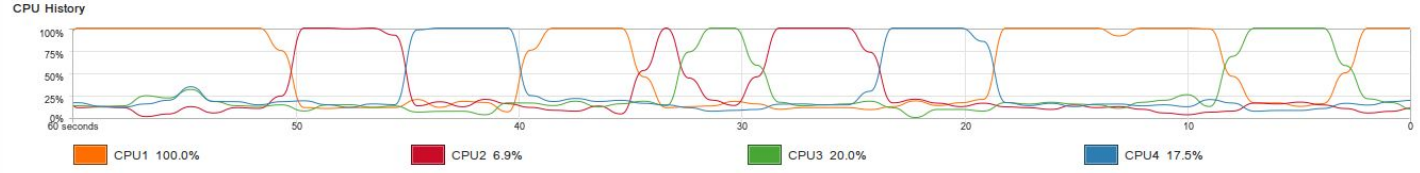
```bash
# Example linear code
# This runs the program one after
#  the other 200 times

# for loop
for num in {1..200}
do
# generate input/output variables
foo="in$num.txt"
bar="shout/out$num.txt"

# run multdiv
multdiv < $foo > $bar

# loop finished
done
exit 0

```

Parallel code: Swift script

```
type file;

app (file out) md (file input) {
    multdiv stdin=filename(input) stdout=filename(out);
}


foreach i in [1:200] {
    file fin <single_file_mapper; file=strcat("in" + i + ".txt")>;
    file f <single_file_mapper; file=strcat("output" + i + ".txt")>;
    f = md(fin);
}

```

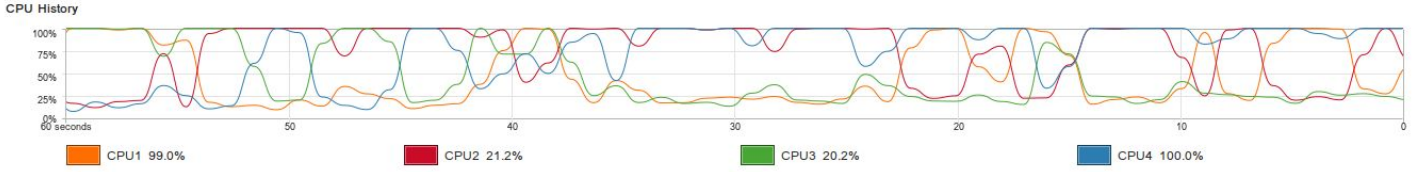
Example

Linear code




```
real    17m41.983s
```

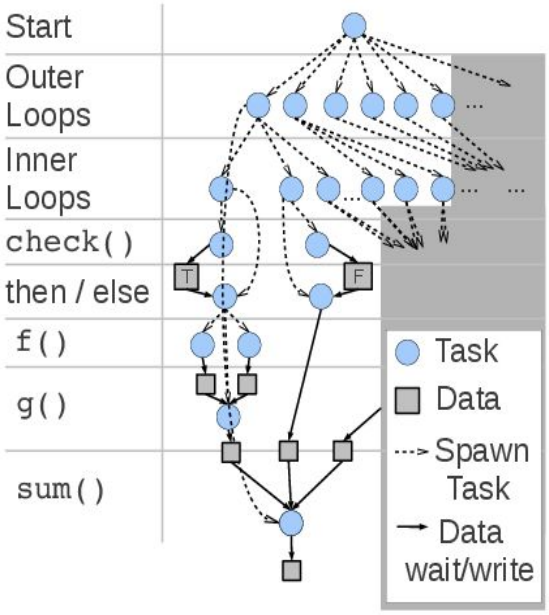
Parallel code




```
real    10m54.748s
```

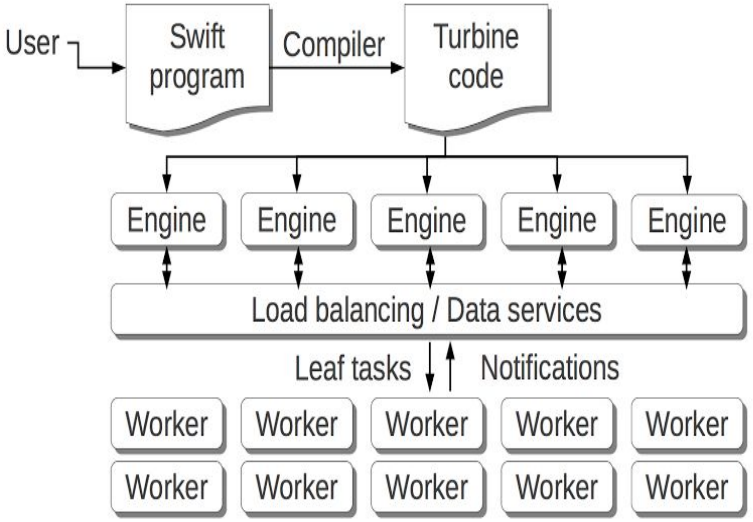
Swift /T

- Increased Number of tasks (Over 1 billion/s supported).
- Ability to execute scripts in embedded interpreters from popular scripting languages.
- Improved built in libraries such as string or math.
- **Runs as a single MPI job.**
- Ability to call native code functions from C, C++, and Fortran.



/T for Turbine

- Swift uses a single compute node. This leads to scalability problems as large numbers of tasks are generated.
- Swift /T seeks to subvert this problem using its distributed Turbine engine.
- The swift compiler divides up tasks into leaf or control tasks which are managed by the Turbine at runtime



References

Main Web Page

Swift-lang.org

Swift /T Performance and Implementation

swift-lang.org/papers/pdfs/Turbine_2013.pdf