John Zhang - Project Manager
Jordan Rupprecht - Systems Architect
Sharath “Language” Gururaj
Philip Tjimos - Verification & Testing
Ad astra per alia porci

to the stars on the wings of a pig
The Problem

• Want to run a program on many independent inputs?
• Don’t want to waste your own processor power?
• Want to use your computer for something else while your work is being executed?
The Solution

• Distributed computing!
• Pigasus allows you to utilize the power of networked machines to run arbitrary programs.
• Existing work: Hadoop, MapReduce.
• But those require expensive infrastructure and setup!
Who would use Pigasus?

- Students
  - e.g. CSEE4823 - Computer Arch.
- Researchers
- Small businesses on a budget
Language Properties

- simple, easy to code
- C, C++ like
- system integrated
- “distributed”
job compile(list inputs) {
  @{"gcc -c %s" % ((string)inputs[0])};
}

void main(list args) {
  list my_servers = [ [ "host"=>"localhost", "user"=>"dummy", "password"=>"dummypw" ] ];

  connect my_servers;

  list srcs = [[(file)"a.c"], [(file)"b.c"], [(file)"c.c"], [(file)"d.c"]];  

  list output;
  int distcompile = push compile, null, srcs, output;
  wait distcompile;

  string ld = "gcc -o MyCoolProgram";

  for (int i = 0; i < length output; i = i + 1) {
    map out = (map)output[i];
    for (int j = 0; j < length (list)out["files"]; j = j + 1) {
      ld = "%s %s/%s" % [ld, (string)out["root"], (string)((list)out["files"])[j]];
    }
  }

  @{ld};
}
void compile( List inputs ) {
    System ( FormatString("gcc -c %s", NewList( CastToString( inputs [ 0 ] ) ) ) ) ;
}

int main(int argc, char** argv) {string buffer;
    ReadStringFromFile(string(argv[1]), buffer);
    List *inputs = UnserializeList(buffer);
    compile(*inputs);
    delete inputs;
    return 0;
}

void PigMain ( List &args ) {
    List my_servers = NewList( Map( "host","localhost" ) + Map( "user","dummy" ) + Map( "password","dummypw" ) ) ;
    global_servers->SetServers(my_servers);
    List srcs = NewList( NewList( CastToFile( "a.c" ) ) + NewList( NewList( CastToFile( "b.c" ) ) ) + NewList( NewList( CastToFile( "c.c" ) ) ) + NewList( NewList( CastToFile( "d.c" ) ) ) ) ;
    List output ;
    int distcompile = Push ( "compile" , List() , srcs , &output ) ;
    Wait ( distcompile ) ;
    string ld = "gcc -o MyCoolProgram" ;
    for ( int i = 0 ;
        i < GetLength( output ) ;
        i = i + 1 ) {
        Map out = CastToMap( output [ i ] ) ;
        for ( int j = 0 ;
            j < GetLength( CastToList( out [ "files" ] ) ) ;
            j = j + 1 ) {
            ld = FormatString("%s %s/%s", NewList( ld ) + NewList( CastToString( out [ "root" ] ) ) + NewList( CastToString( ( CastToList( out [ "files" ] ) ) [ j ] ) ) ) ;
        }
    }
    cout << ( ( ld + "\n" ) ) ;
    System ( ld ) ;
    while (true) {
        bool a = job_thread_pool->HasJobsRunning();
        bool b = send_thread_pool->HasJobsRunning();
        bool c = get_thread_pool->HasJobsRunning();
        if (!(a || b || c)) break;
        sleep(1);
    }
}
Executing distcc

distcc.pig
pigasus compiler
pig.out compile.out

gcc -o MyCoolProgram *.o
System Architecture

Main

PigMain()

Job Threadpool

Send Threadpool

Get Threadpool

input files
executables
jobs

inputs[0]
inputs[n]

result files

push

wait

TIME
Development Environment

- Front-end developed in OSX/Windows.
- Back-end developed in Linux.
- Tools used:
  - Lex/Bison
  - C++, Bash, Expect
  - Google Code (SVN)
Runtime Environment

- Host machine:
  - POSIX, Bash, Expect, g++, SSH.

- Remote machines:
  - Same architecture and OS as host.
  - Bash, SSH server.
Testing Strategy

- Unit Testing
- Test each of the ~120 productions in our grammar.
- We want to test different servers, failing servers, etc.
Conclusion

• We probably should’ve used Java...
• Our language is much more complex than we originally envisioned.
• Threads are cumbersome.