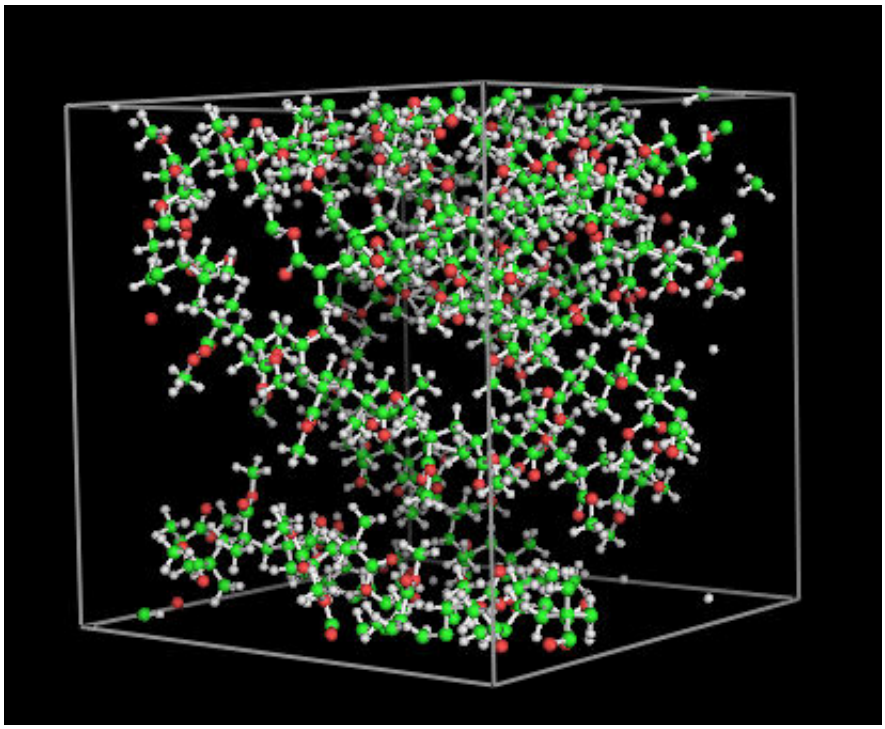
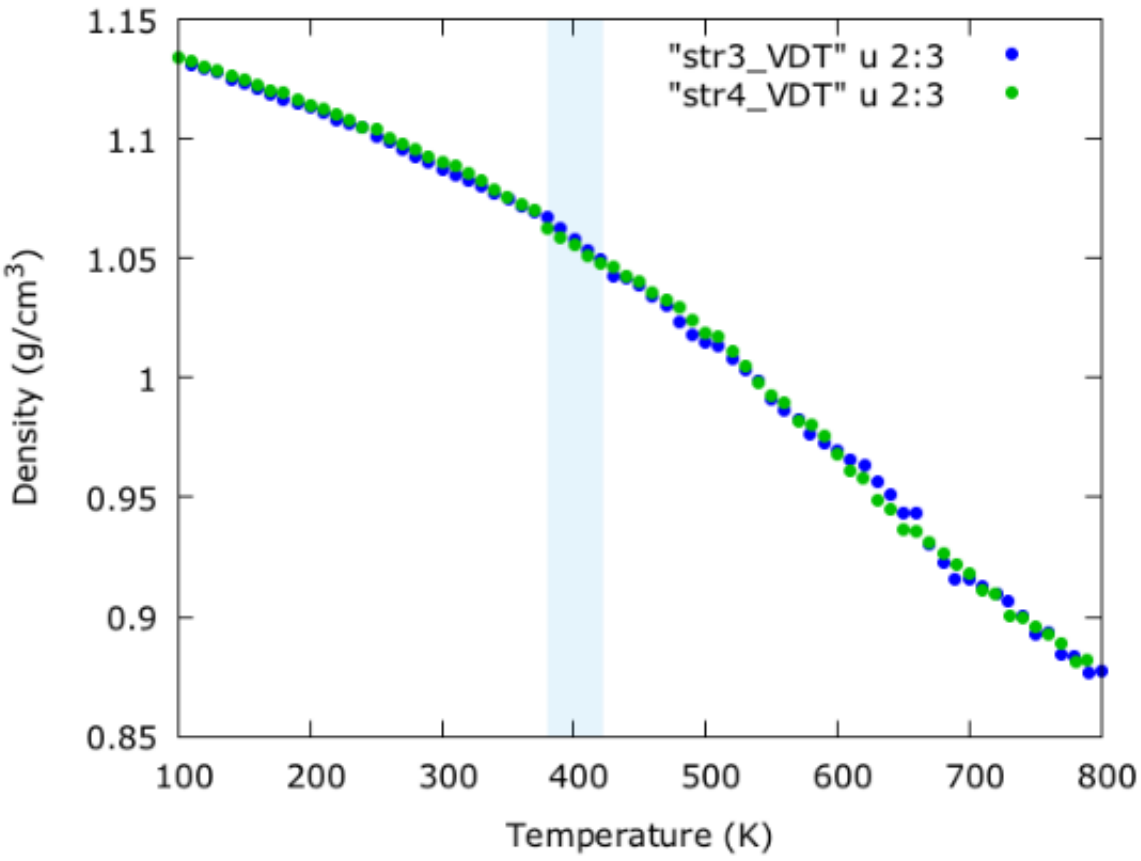
Polymer Service

Calculation of the glass transition temperature of an amorphous system

Overview

- Build the atomistic system (e.g. PolymerModeler)
- Run LAMMPS: cool the system and shrink the box
- Analyze the output from LAMMPS: extract density vs. T





(Image courtesy Lorena Alzate)

Service proposal

- GET info on available services returns JSON

```
{
   "polybuild": {
      "input": {
         "values": {
            "monomer": {
               "type": "string",
               "desc": "Name of the basic repeating unit",
               "values": ["pmma", "pe", "ps"]
            },
            "chains": {
               "type": "integer",
               "desc": "Number of chains to build"
            },
            "monomers": {
               "type": "integer",
               "desc": "Number of monomers in each chain (polydispersi
```

```
ty)"
            },
            "interactions": {
               "type": "string",
               "desc": "Name of interactions between particles",
               "values": ["LJ"]
            },
            "range": {
               "type": "real",
               "units": "Angstroms",
               "desc": "Interaction range between particles"
            },
            "density": {
               "type": "real",
               "units": "g/cc",
               "desc": "Density of the constructed amorphous system"
            },
            "temperature": {
               "type": "real",
               "units": "K",
               "desc": "Temperature of the system while building"
            },
            "configurations": {
               "type": "integer",
               "desc": "Number of configurations per torsion during MC
 sampling",
            },
            "rotate_torsions": {
               "type": "string",
               "desc": "Specify how many torsions in each monomer are
rotated",
               "values": ["all", "only_between"]
            },
            "torsion_probs": {
               "type": "string",
               "desc": "Specify how torsion angles are assigned",
               "values": ["random", "fixed", "sp2", "sp3"]
            },
            "forcefield": {
               "type": "string",
               "desc": "Name of the forcefield to apply for LAMMPS dat
a file",
               "values": ["Dreiding"]
            }
         }
      },
```

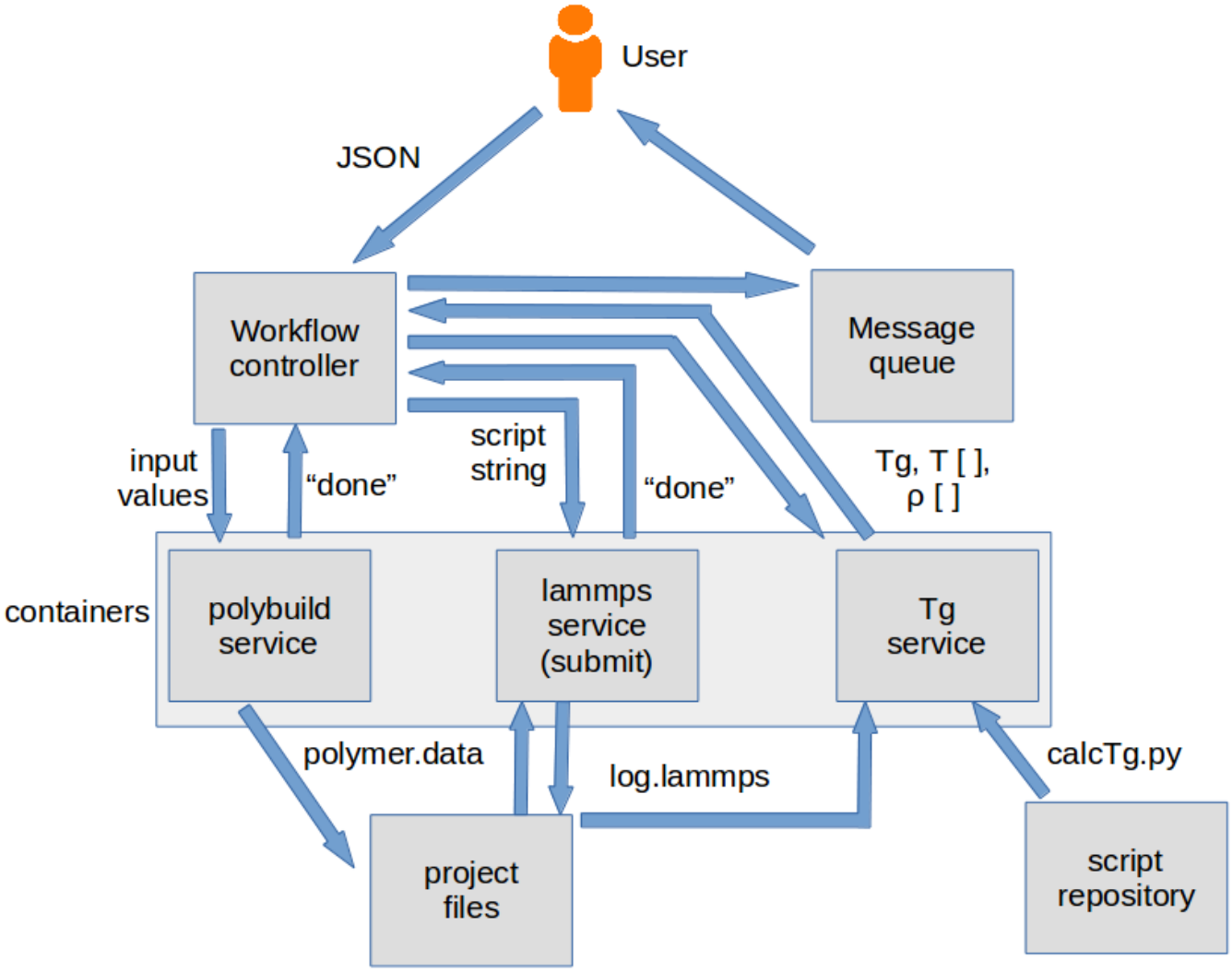
```
      "output": {
         "files": ["lammps_data_file"]
      }
   },

   "lammps": {
      "input": {
         "files": ["lammps_script", "lammps_data_file"]
      },
      "output": {
         "files": ["lammps_log_file"]
      },
      "submit_options": {
         "cores": {
            "type": "integer",
            "desc": "Number of cores on which to run LAMMPS"
         },
         "walltime": {
            "type": "real",
            "units": "hours",
            "desc": "Requested simulation time for LAMMPS"
         }
      }
   }

   "Tg": {
      "script": "calcTg.py",
      "input": {
         "files": ["lammps_log_file"]
      },
      "output": {
         "values": {
            "Tg": {
               "type": "real",
               "units": "K",
               "desc": "Glass transition temperature"
            }
            "density": {
               "type": "array",
               "units": "g/cc",
               "desc": "System density over MD steps"
            },
            "temperature": {
               "type": "array",
               "units": "K",
               "desc": "System temperature over MD steps"
```

```
            }
        }
    }
  }
}
```

- Construct JSON input describing a workflow
  - polybuild: build the atomistic system
  - lammps: run LAMMPS to cool and compress
  - Tg: identify Tg from LAMMPS thermo output




```
{
   "tasks": ["polybuild", "lammps", "Tg"],
```

```
  "polybuild": {
     "input": {
        "values": {
           "monomer": "pmma",
           "chains": 10,
           "monomers": 5,
           "interactions": "LJ",
           "range": 6.0,
           "density": 0.5,
           "temperature": 300,
           "configurations": 50,
           "rotate_torsions": "all",
           "torsion_probs": "random"
           "forcefield": "Dreiding"
        }
     },
     "output": {
        "files": {
           "lammps_data_file": "polymer.data"
        }
     }
  },

  "lammps": {
     "input": {
        "files": {
           "lammps_script": {
              "contents": "units real\natom_style full\nboundary p p p\nspe
cial_bonds lj/coul 0.0 0.0 1.0 dihedral yes\ndielectric 1.0\npair_style lj/c
ut 6.0\nbond_style harmonic\nangle_style harmonic\ndihedral_style harmonic\n
improper_style harmonic\nread_data polymer.data\nneighbor 2.0 bin\nthermo_st
yle custom step etotal ke temp pe ebond eangle edihed eimp evdwl ecoul elong
 press pxx pyy pzz pxy pxz pyz lx ly lz\nthermo 50\nthermo_modify flush yes\
nMINIMIZE\ndump 1 all custom 1 polymer.dump id mol type xu yu zu\ndump_modif
y 1 format \"%d %d %d %.10f %.10f %.10f\"\nrun 0\nundump 1\nwrite_restart po
lymer.restart\n"
           }
           "lammps_data_file": "polymer.data"
        }
     },
     "output": {
        "files": {
           "lammps_log_file": "log.lammps"
        }
     },
```

```
    "submit_options": {
       "cores": 4,
       "walltime": 12.0
    }
  }
}
```