Overview and Motivation

- Auto-tuning has a broad range of applications
  - Optimization of run-time, throughput, energy, etc.
- Supporting new features increasingly difficult due to monolithic internals
  - 10+ years of non-structured development
  - Core search logic implemented in Tcl
- Reorganization produced a generalized auto-tuning framework
  - Á la componentization of Dyninst API
  - Eases integration with other projects
Auto-tuning Basics

• Empirically test configurations for performance
  o Searches a set of valid configurations for optimal

• Valid configurations defined by tuning variables
  o Variables can be integer, real, or enumerated strings
  o Each variable bound by minimum, maximum, and step

• Set of tuning variables define a parameter space
  o N variables create an N-dimensional space
  o Cartesian coordinates represent unique configurations
Parameter Space Example

- **Variable 1**
  - Name: Tile
  - Min: 2
  - Max: 16
  - Step: 2

- **Variable 2**
  - Name: Unroll
  - Min: 0
  - Max: 10
  - Step: 1
Auto-Tuning Feedback Loop

- Search Strategy
- Evaluated Performance
- Candidate Points
- Client Application
Search Strategies

• Drives the feedback loop

• Pluggable module with two major interfaces
  o Fetch: Generates a new point for client
  o Report: Accepts a point/performance pair from client

• Four strategies included in Active Harmony
  o Brute Force
  o Random
  o Nelder-Mead
  o Parallel Rank Order (PRO)
Generalized Auto-Tuning

Candidate Points

 Evaluated Performance

Search Strategy

Client Application

FETCH

REPORT

1 2 3
Onion Model Workflow

- Allows for paired functionality, but either hook is optional.
- Fetch hooks executed in ascending order.
- Report hooks executed in descending order.
- Point values cannot be modified (directly).
Plug-in Workflow: ACCEPT

- Indicates successful processing at this level.
- Plug-in relinquishes control of the point to entity below.
Plug-in Workflow: RETURN

- Allows a short-circuit in the feedback loop.
- Plug-in must provide the performance value.
- Processing resumes at the same level from the report workflow.
Plug-in Workflow: REJECT

- Allows modification of the search indirectly.
- Plug-in must provide an alternate valid point.
  - Search strategy is not obligated to use it.
- Processing jumps directly back to search strategy.
Plug-in Example: Point Logger

```c
logger_init()
{
    outfd = open("performance.log");
}

logger_report(flow_t *flow, point_t *pt, double perf)
{
    fprintf(outfd, "%s -> %lf
", string(pt), perf);
    flow->type = ACCEPT;
}

logger_fini()
{
    close(outfd);
}
```
Plug-in Workflow: WAIT

- Allows asynchronous point processing
- Plug-in sends point to external process
- Other points processed in the mean time
- Processing resumes at the same level upon return
Plug-in Example: Code Server

cs_init()
{
    sockfd = tcp_connect(CODE_SERVER);
    register_fetch_callback(sockfd, code_ready);
}

cs_fetch(flow_t *flow, hpoint_t *pt)
{
    send_point(sockfd, pt);
    flow->type = WAIT;
}

code_ready(flow_t *flow, hpoint_t *pt_list[])
{
    recv_completed_id(sockfd, id);
    flow->point = find_point_in_list(pt_list, id);
    flow->type = ACCEPT;
}
Plug-in Example: Constraints

• Support for non-rectangular parameter spaces.
  o Implemented as plug-in #2 using REJECT workflow.
  o $y \leq x$
Plug-in Example: TAUdb

- Integration with TAU via TAUdb.
  - Tuning and Analysis Utilities
- Search data stored in persistent database

- Adds parameter analysis to any tuning session.
- Can potentially be used as result cache
Specifying Plug-ins

• Pluggable modules read from configuration
  o Unstructured key/value pair system queried at launch

```bash
> cat harmony.cfg
SESSION_STRATEGY=pro.so
SESSION_PLUGIN_LIST=logger.so:taudb.so
```

• Directly from the command-line
  o Tuna to accept in-line configuration directives
  o Syntax to be determined
Availability

• Current release is Active Harmony 4.0
  o Includes Tuna and preliminary plug-in interface

• Upcoming release to include
  o Improved plug-in interface
  o Library of basic strategies and plug-ins
  o No more Tcl code!
Summary and Next Steps

• Generalized auto-tuning framework presented
  o Variety of systems possible with minimal coding
  o Functionality encapsulated to simplify development

• Next steps
  o Develop library of search strategies and plug-ins
    o Hierarchical search?
    o We could use your help!
  o Investigation of further abstractions
    o Parameter spaces