Overview

- Autotune goals and partners
- Periscope Tuning Framework (PTF)
- MPI Parameters Plugin
- Demo
Goals

• Tackle complexity of HPC architectures
  – Multicore, multisocket, accelerators, DVFS
• Enable higher productivity via auto-tuning
• Focus on static tuning in pre-production phase
  – Produce tuning recommendations
• Leverage state of the art performance analysis
• Implement an extensible environment
  – Support open and proprietary plugins
Partners

- UNIVIE
  - High level Parallel Patterns for GPGPU Plugin
- CAPS
  - Hybrid Manycore Tuning: HMPP Codelet Tuning Plugin
- LRZ
  - Energy Consumption via CPU Frequency Tuning Plugin
- UAB
  - Master-Worker MPI Plugin
  - MPI Runtime Plugin
- TUM
  - Compiler Flag Selection Plugin
  - User-level Tuning Plugin
- ICHEC
Overview

• Autotune partners and goals

• Periscope Tuning Framework (PTF)

• MPI Parameters Plugin

• Demo
Periscope

• On-line
  -- no need to store trace files

• Distributed
  – reduced network overhead
  – based on autonomous cooperating agents

• Analyzes:
  – MPI Communication
  – Single-node Performance
  – OpenMP Performance

• Supports: Fortran, C/C++
Periscope Tuning Framework

PTF

- Extension of Periscope
- Online tuning process
  - Application phase-based
- Extensible via tuning plugins
  - Single tuning aspect
  - Combining multiple tuning aspects
- Rich framework for plugin implementation
- Automatic and parallel experiment execution
Terminology

• Tuning Parameter
  – Name + integer range

• Tuning space $TS_P$ of plugin $P$
  – $TS_P = TP_1 \times TP_2 \times \ldots \times TP_k$

• Variant space $VS_r$ of region $r$
  – subset of tuning space
  – variant $v = (v_1, \ldots, v_k)$

• Objective
  – $obj: REG_{appl} \times TS_P \rightarrow \mathbb{R}$

• Search Space
  – $((r_1, \ldots, r_n), VS, \{obj_1, \ldots, obj_n\})$
Terminology

- **Tuning scenario** $\text{sc}_r$ of region
  - $\text{sc}_r = ((r_1, \ldots, r_n), (v_1, \ldots, v_k), \{\text{obj}_1, \ldots, \text{obj}_n\})$

- **Tuning action** $\text{TA}_i$
  - Enforces $v_i$ for the tuning point $i$ in variant $\nu$
  - Runtime tuning actions: variable assignment and function call
  - Pre-runtime tuning actions: Compilation, #MPI tasks, MPI parameters

- **Search algorithm**
  - $\forall \nu \in \text{VS} \ \text{create} \ ((r_1, \ldots, r_n), \nu, \{\text{obj}_1, \ldots, \text{obj}_n\})$
  - Multiple search spaces: Scenarios are the xproduct of all scenarios of each search space.
Tuning Plugins

- Extract Parameters from Sir file or Configuration file
- Initialize Tuning Parameters
- Load Dynamic Library with the implementation of Search Algorithm
- Initialize Search Algorithm
- Create Variant Space and add to Search Space
- Assign regions to Search Space
- Assign Search Space to Search Algorithm
- Assign Objectives to Search Algorithm
1. Create Variant
2. Assign Variant + Region + Objective to Scenario
3. Add Scenario to Created Scenario Pool (CSP)
4. If there are more Variants goto 1.
1. Get Scenario from CSP
2. Perform any action required by the Scenario (compile, change environment variables)
3. Add Scenario to Prepared Scenario Pool (PSP)
4. If several Scenarios can be tested in parallel goto 1.
1. Get Scenario from PSP
2. Assign rank to experiment or make it exclusive
3. Add Scenario to the Execution Scenario Pool (ESP)
4. If several Scenarios can be tested in parallel goto 1.
1. Should the application be restarted for running the Experiments

2. Should the command for the execution be changed
Tuning Plugin

More scenarios in the CSP?

More scenarios in the PSP?

New things to test?

Init

Prep Search

Creat Scen

Def Exp

Exec Scen

Prep Scen

Restart App

Run Exp

Proc Results

Creat Advice

Done?

Init

Pgin

Tuning Plugin
Overview

• Autotune partners and goals

• Periscope Tuning Framework (PTF)

• MPI Parameters Plugin

• Demo
MPI Parameters Plugin:  
Introduction

• Goal
  – Reduce time of user region based on finding the right combination of user provided MPI parameters.

• Tuning Parameters
  – Each parameter has a list of values to test that can be indicated explicitly or with a range

• Tuning Action
  – The tuning action is to set a variant of the parameters in the preparation phase of a scenario before reexecuting.
MPI Parameters Plugin: Control Flow

- Initialize MPIParameterTP from a configuration file:
  - Parameter_Name=<comma separated list>
  - Parameter_Name=<initial:step:final>
- Map MPIParameterTP to Tuning Parameters
- Load Dynamic Library with the implementation of the Search Algorithm developed for this Plugin
- Initialize the Search Algorithm
• Generate the xproduct of Parameters’ values
• Exhaustive search for all variants in the variant space
MPI Parameters Plugin:
Control Flow

1. Get Scenario from CSP
2. Map Tuning Parameters to MPIParametersTP
3. Create new command line
4. Add Scenario to Prepared Scenario Pool (PSP)
MPI Parameters Plugin: Control Flow

1. Application should be restarted
2. Use the command line built in the preparation.
MPI Parameters Plugin: Control Flow

1. Retrieve the best variant from the search algorithm
2. Give it as tuning recommendation
Overview

• Autotune partners and goals

• Periscope Tuning Framework (PTF)

• MPI Parameters Plugin

• Demo
**Tuning Model**

- **Tuning Parameter**
  - Type: variable
  - Name: TEST
  - Range: 1..10

- **Tuning space**: TS = 1..10

- **Variant space** = TS

- **Tuning scenario**:
  - ((1, 485), (1), {ExecTime}) to ((1, 485), (10), {ExecTime})

- **Tuning action**
  - Assign 1 to variable \texttt{variant in start\_region}(1, 485)
do k=1,20
   variant=k
   !$MON USERREGION TP name(Test) variable(variant) variants(10)
   tstart=MPI_Wtime()
   mm=5-variant+1
   if (mm<=0)
      m= -1*mm+1
   endif
   call sleep(mm)
   tend=MPI_Wtime()
   write (*,*) myrank, variant, tend-tstart
   !$MON END USERREGION
endo