a massively-parallel heterogeneous computing framework for optimization and parameter sensitivity analysis

Mike McKerns

California Institute of Technology

April 12, 2011

http://dev.danse.us/trac/mystic

Pathos – Trac

http://dev.danse.us/trac/pathos

Google | tmp | My Software | Music | News | Popular | CIT IMSS | CIT Webmail | Python | MacPorts | arXiv.org | LaTeX | PyMOTW | Training

Search

Login | Settings | Help/Guide | About Trac

Wiki | Timeline | Roadmap | Browse Source | View Tickets | Search | Project List

Start Page | Index by Title | Index by Date | Last Change

pathos: a framework for heterogeneous computing

|| User Guide || Download || Tutorials || Manual || License || Feedback ||

About Pathos

Pathos is a framework for heterogenous computing. It primarily provides the communication mechanisms for configuring and launching parallel computations across heterogenous resources. Pathos provides stagers and launchers for parallel and distributed computing, where each launcher contains the syntactic logic to configure and launch jobs in an execution environment. Some examples of included launchers are: a queue-less MPI-based launcher, a ssh-based launcher, and a multiprocessing launcher. Pathos also provides a map-reduce algorithm for each of the available launchers, thus greatly lowering the barrier for users to extend their code to parallel and distributed resources. Pathos provides the ability to interact with batch schedulers and queuing systems, thus allowing large computations to be easily launched on high-performance computing resources. One of the most powerful features of pathos is "tunnel", which enables a user to automatically wrap any distributed service calls within a ssh-tunnel.

Pathos is divided into four subpackages::

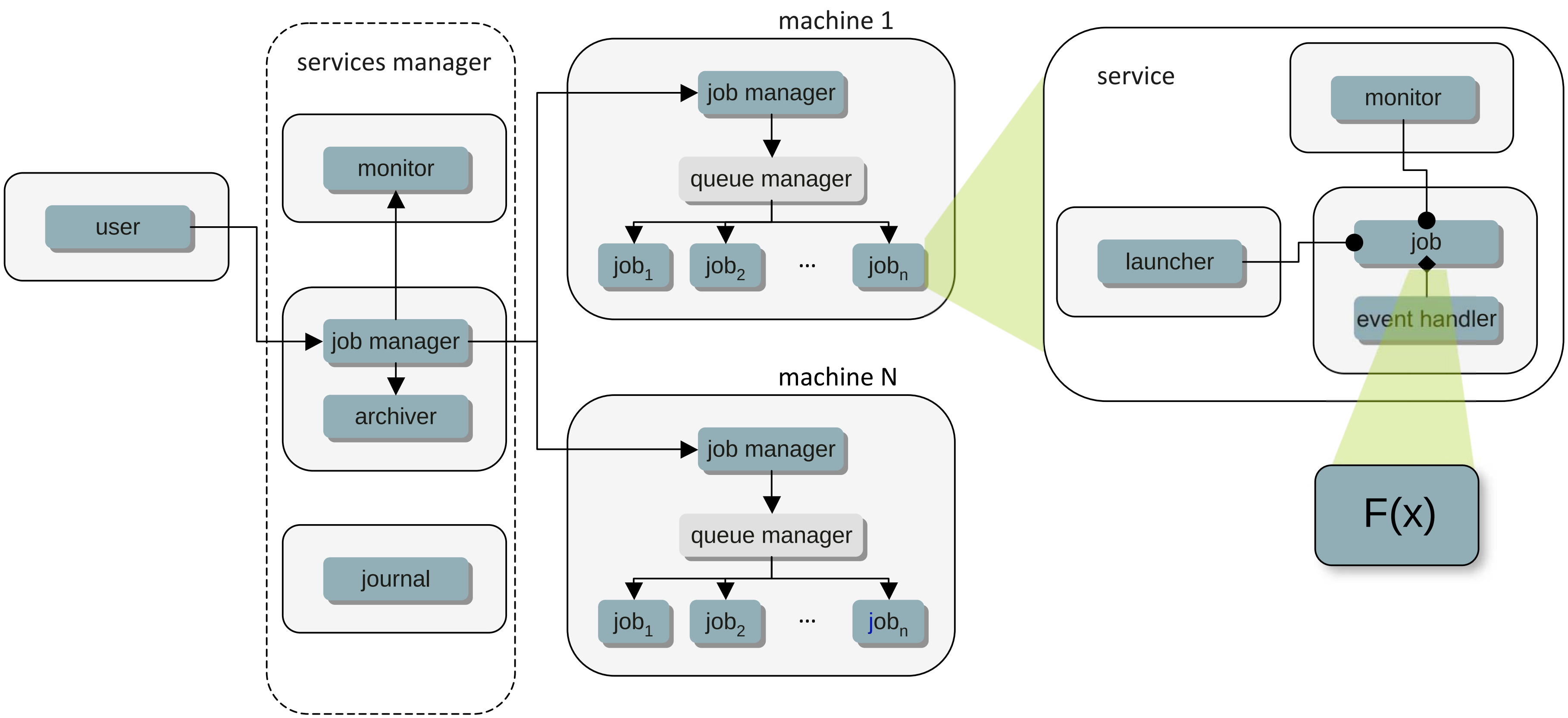
- dill: a utility for serialization of python objects
- pox: utilities for filesystem exploration and automated builds
- pyina: a MPI-based parallel mapper and launcher
- pathos: distributed parallel map-reduce and ssh communication

---

Pathos Subpackage

The pathos subpackage provides a few basic tools to make distributed computing more accessable to the end user. The goal of pathos is to allow

heterogeneous service deployment



goal: make the user's configuration, deployment, and management of F(x) on heterogeneous resources as easy as possible

massively parallel resources

CALTECH PSAAP

hera@llnl: 12,800 opteron cores

lobo@lanl:4352 opteron cores

coyote@lanl:2676 opteron cores

shc@cacr: 1180 opteron cores

distributed parallel computing

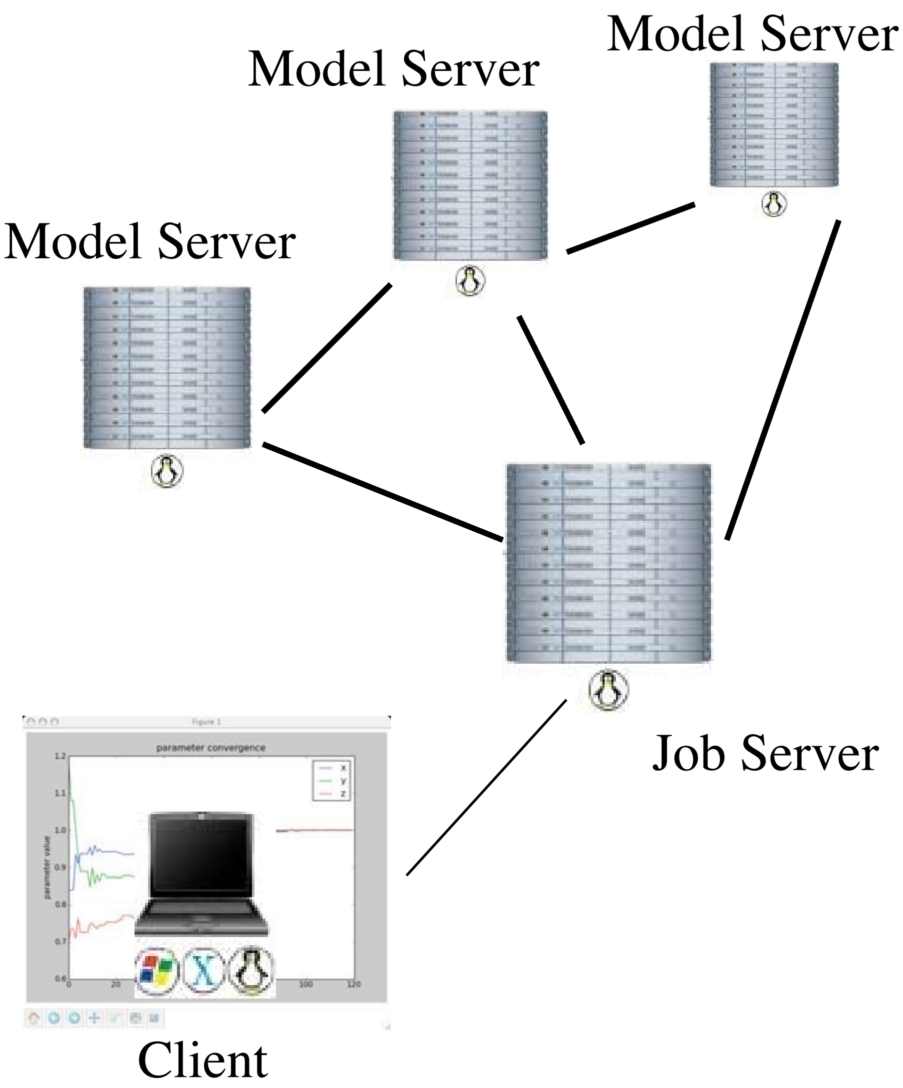
- Parallel computing:

  - Each model may require it's own parallel compute cluster.

- Distributed parallel:

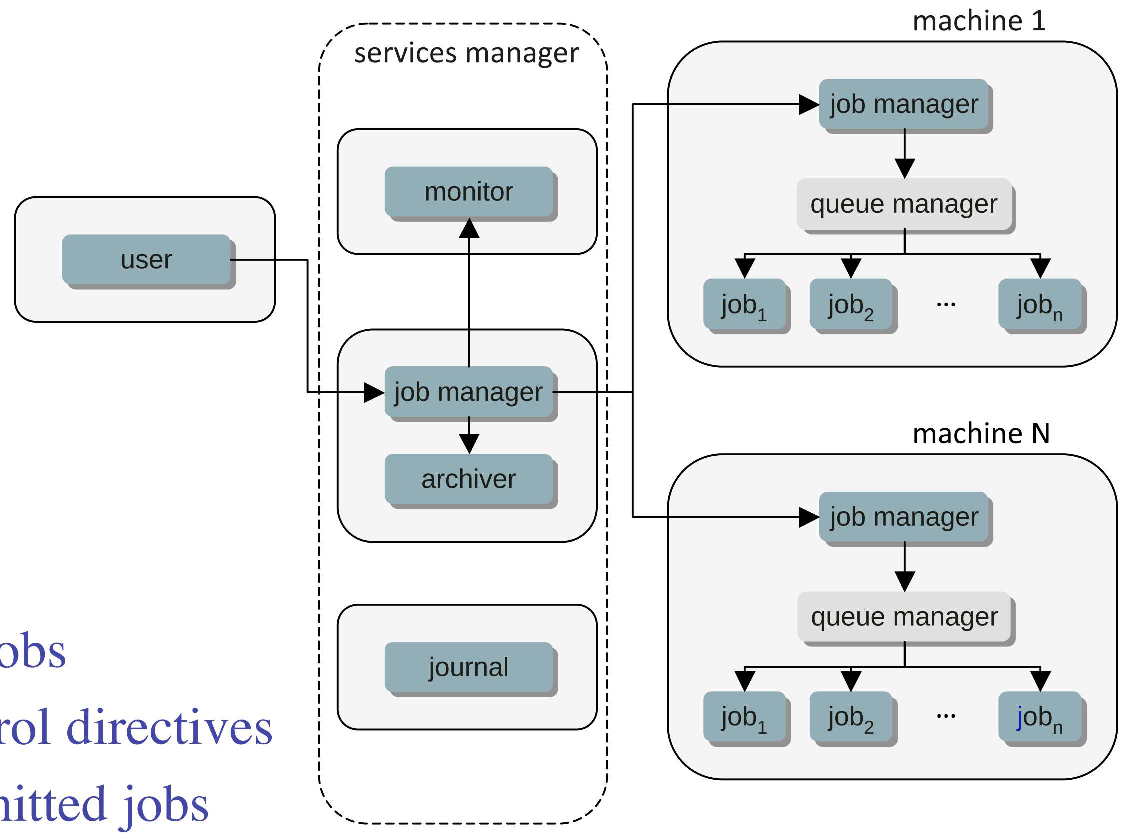
  - Multiple copies of models working together for global optimization and in parameter sensitivity studies.

  - Queue configuration and launch commands must be passed between compute resources.



distributed job management

- The goal is to abstract the mechanism used in parallel job queue managers.



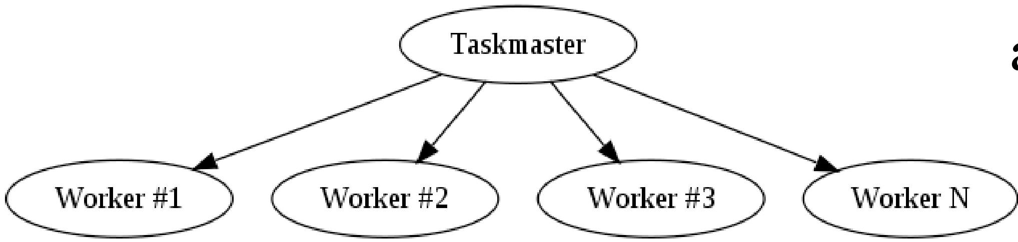
- the Job Manager
  - stages and launches new jobs
  - broadcasts execution control directives
  - maintains registry of submitted jobs

- the Job (or worker)
  - reacts to control directives

a "job" is the fundamental commodity

we abstract all of out jobs as "services"

managers & workers as services

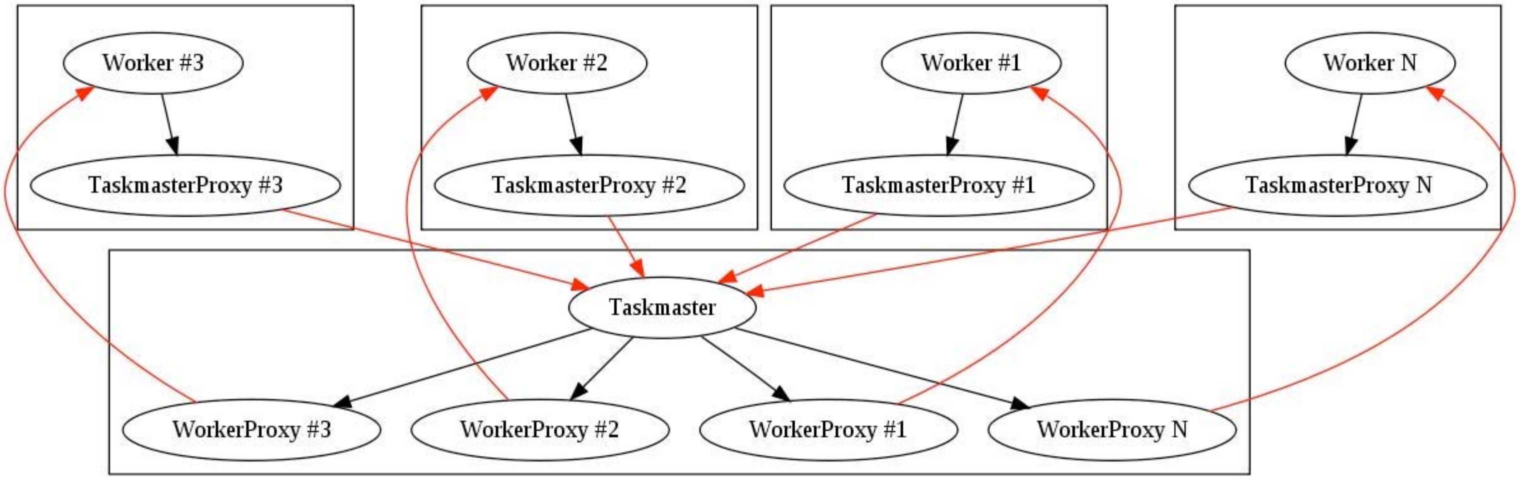


The Taskmaster and its Workers

managers & workers abstracted as services, each with a local copy of the computation

use proxies to decouple from networking

proxies insulate "sending" but not "receiving"…

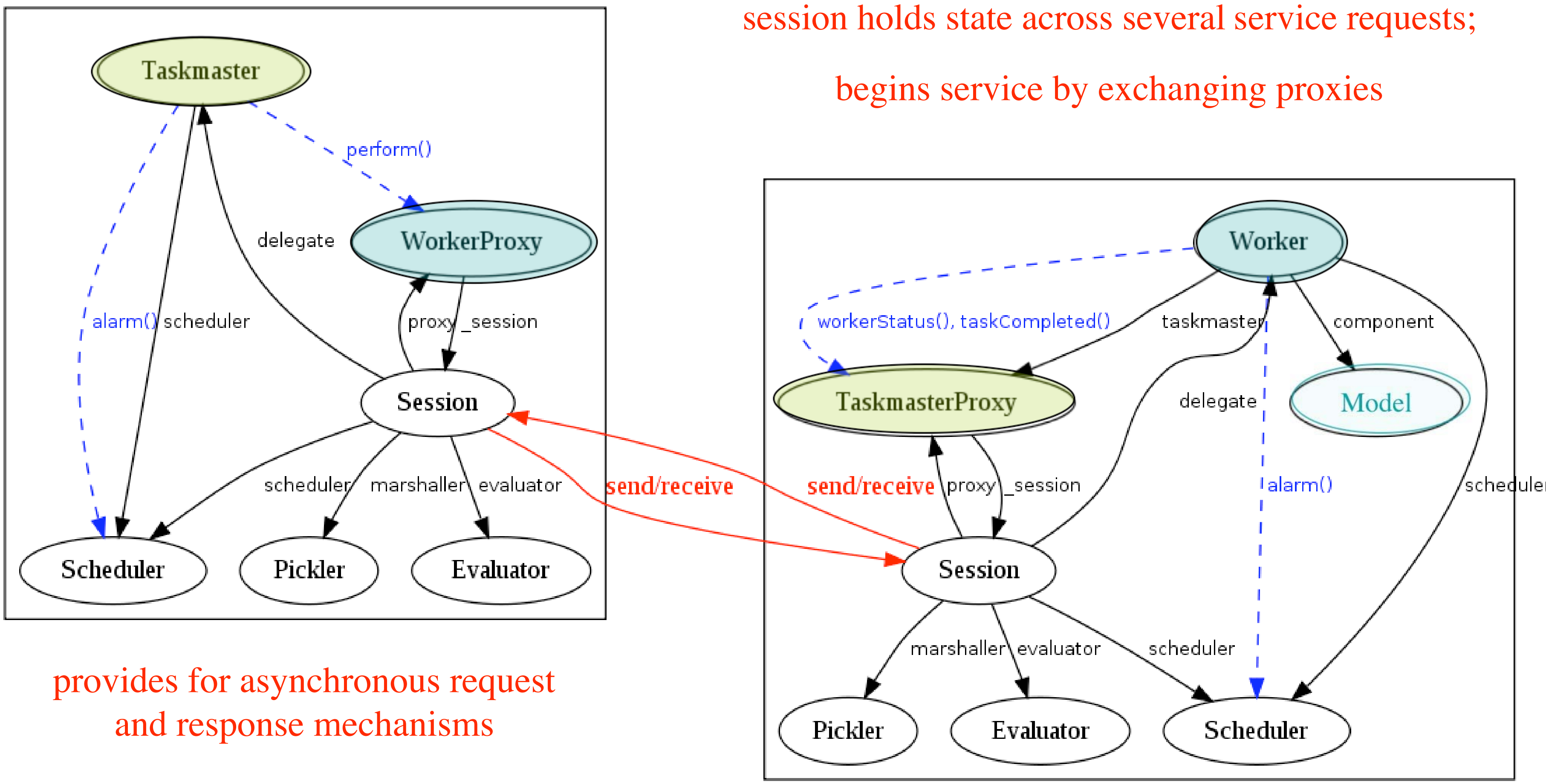


Proxies

networking by proxies & delegates

session holds state across several service requests;

begins service by exchanging proxies



provides for asynchronous request
and response mechanisms

A Single Taskmaster/Worker Session

scheduling and robustness

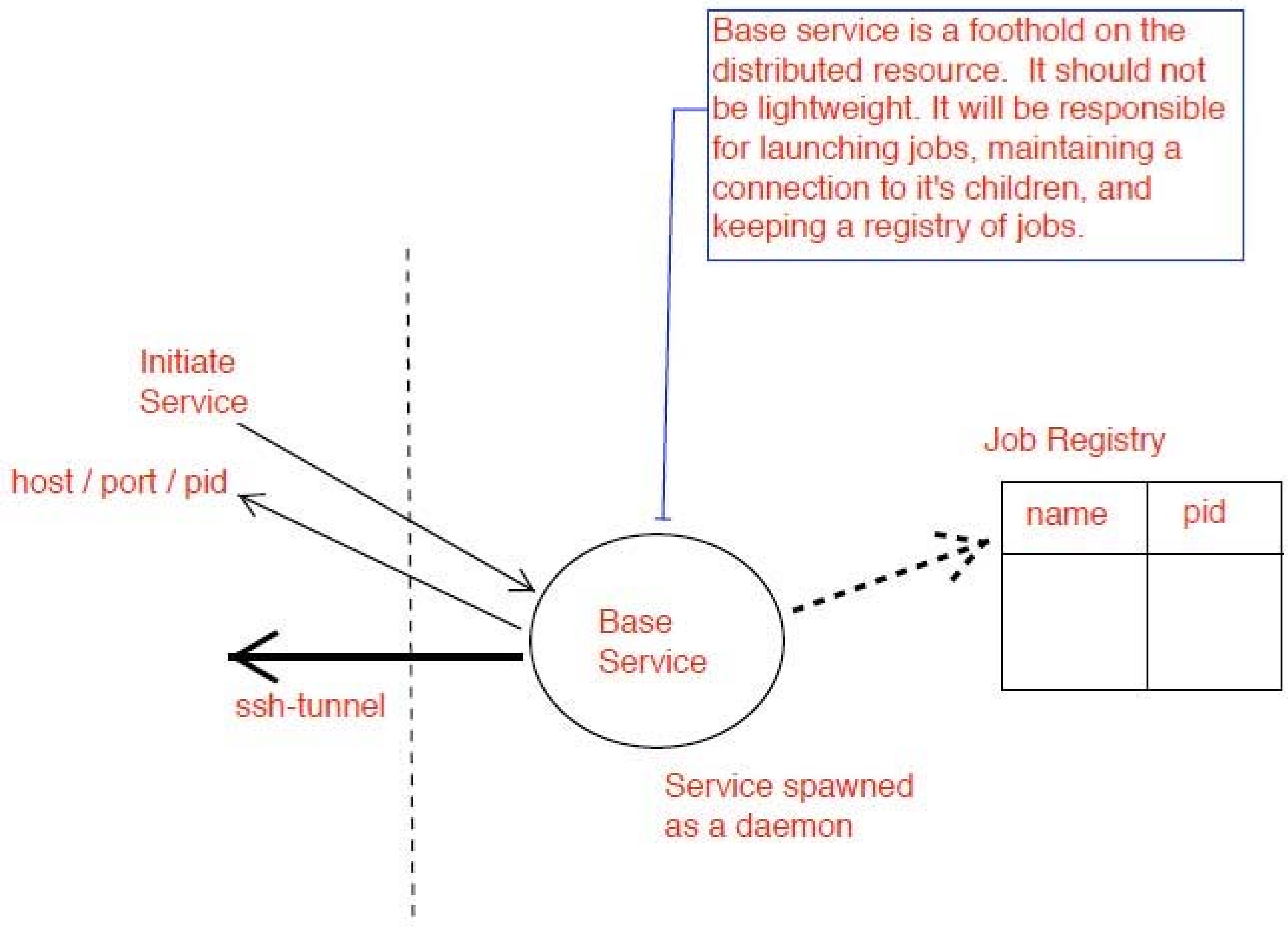
CALTECH PSAAP

- Scheduling encapsulates key responsibilities:
  - Callbacks need to fire at a certain time regardless of how much activity there is. For instance, the `Taskmaster` needs to "fire" any worker it hasn't heard from in a while. Simply checking this in `onTimeout()` is not good enough. What if there is a lot of socket activity from other workers? Then `onTimeout()` never gets called, and the "fire" logic starves.
  - There is no single, fixed/configurable timeout value; it is dynamic.

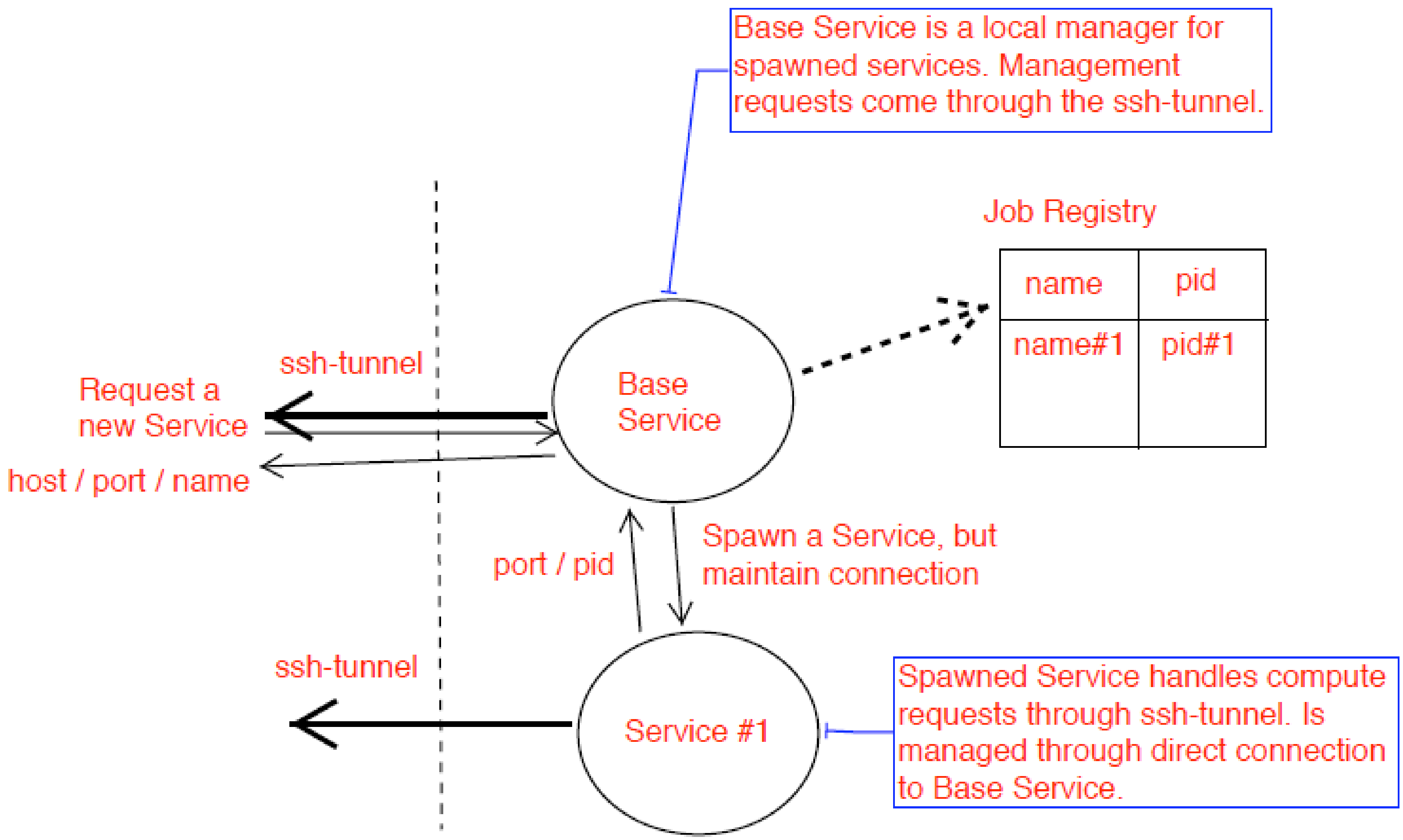
- Robustness required whenever something “goes wrong”
  - monitoring life status of a job
  - restart job from last known state
  - failure recovery is detection, logic, and rebuilding services

Now we must abstract away how services are launched locally…

local deployment of service factory

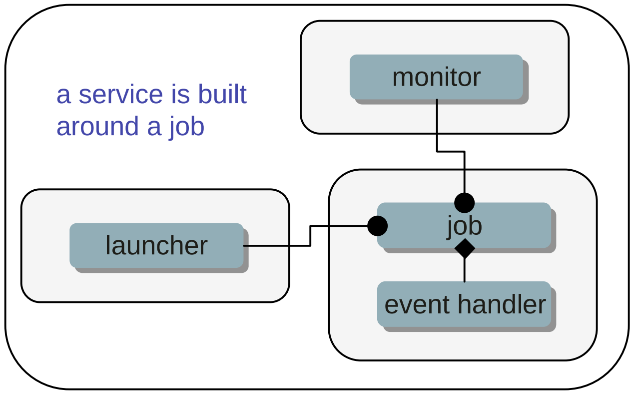


services are spawned on request



services at the “local” level

- launchers submit jobs as services in the current execution environment



- infrastructure enabling configuration of processors, nodes, and clusters into higher-level analysis circuits

- tools for heterogeneous computing
  - dynamic workload balancing or static workload distribution strategy
  - secure ssh-tunneled service communication
  - parallel map interface for strategies

- compound services can be built manually or with a coupling strategy

- available launchers:
  - multiprocessing
  - MPI-based
  - RPC-based

- available schedulers:
  - torque, slurm, lsf

built-in dynamic service coupling

- pathos infrastructure is utilized to build powerful new optimization algorithms
  - global optimization via *N* parallel gradient optimizations over parameter space, each with different initial conditions
  - *N* particle-based stochastic optimizers that seek the global optimum in parallel, each applying a penalty at a candidate optimum
- build compound models and optimizers
  - asynchronously coupled models
  - multi-scale models
  - parallel and nested optimizers
  - dynamic updates of constraints

- mystic builds an optimization framework on top of an infrastructure for heterogeneous computing
  - optimizers, models, and constraints are all callable services
  - services can be connected across distributed and parallel resources
  - services can be connected in series, parallel, nested, etc.

http://dev.danse.us/trac/mystic

Google | tmp | My Software | Music | News | Popular | CIT IMSS | CIT Webmail | Python | MacPorts | LaTeX Editor | PyMOTW | Training

Search

Login | Settings | Help/Guide | About Trac

Wiki | Timeline | Roadmap | Browse Source | View Tickets | Search | Project List

Start Page | Index by Title | Index by Date | Last Change

mystic: a simple model-independent inversion framework

|| User Guide || Download || Tutorials || Manual || License || Feedback ||

About Mystic

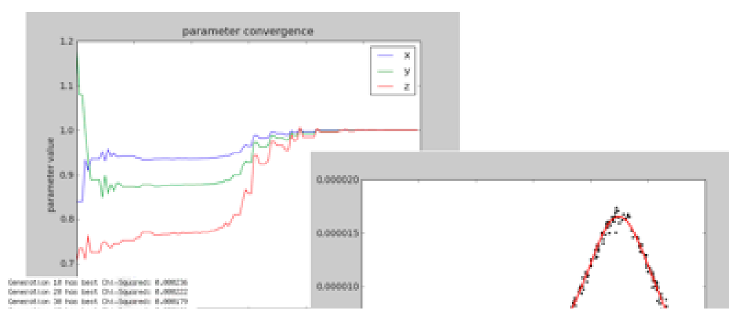
The mystic framework provides a collection of optimization algorithms and tools that allows the user to more robustly (and readily) solve optimization problems. All optimization algorithms included in mystic provide workflow at the fitting layer, not just access to the algorithms as function calls. Mystic gives the user fine-grained power to both monitor and steer optimizations as the fit processes are running.

Where possible, mystic optimizers share a common interface, and thus can be easily swapped without the user having to write any new code. Mystic solvers all conform to a solver API, thus also have common method calls to configure and launch an optimization job. For more details, see `mystic.abstract_solver`. The API also makes it easy to bind a favorite 3rd party solver into the mystic framework.

By providing a robust interface designed to allow the user to easily configure and control solvers, mystic reduces the barrier to implementing a target fitting problem as stable code. Thus the user can focus on building their physical models, and not spend time hacking together an interface to optimization code.

Mystic is in the early development stages, and any user feedback is highly appreciated. Contact Mike McKerns [mmckerns at caltech dot edu] with comments, suggestions, and any bugs you may find. A list of known issues is maintained at http://dev.danse.us/trac/mystic/query.

Major Features

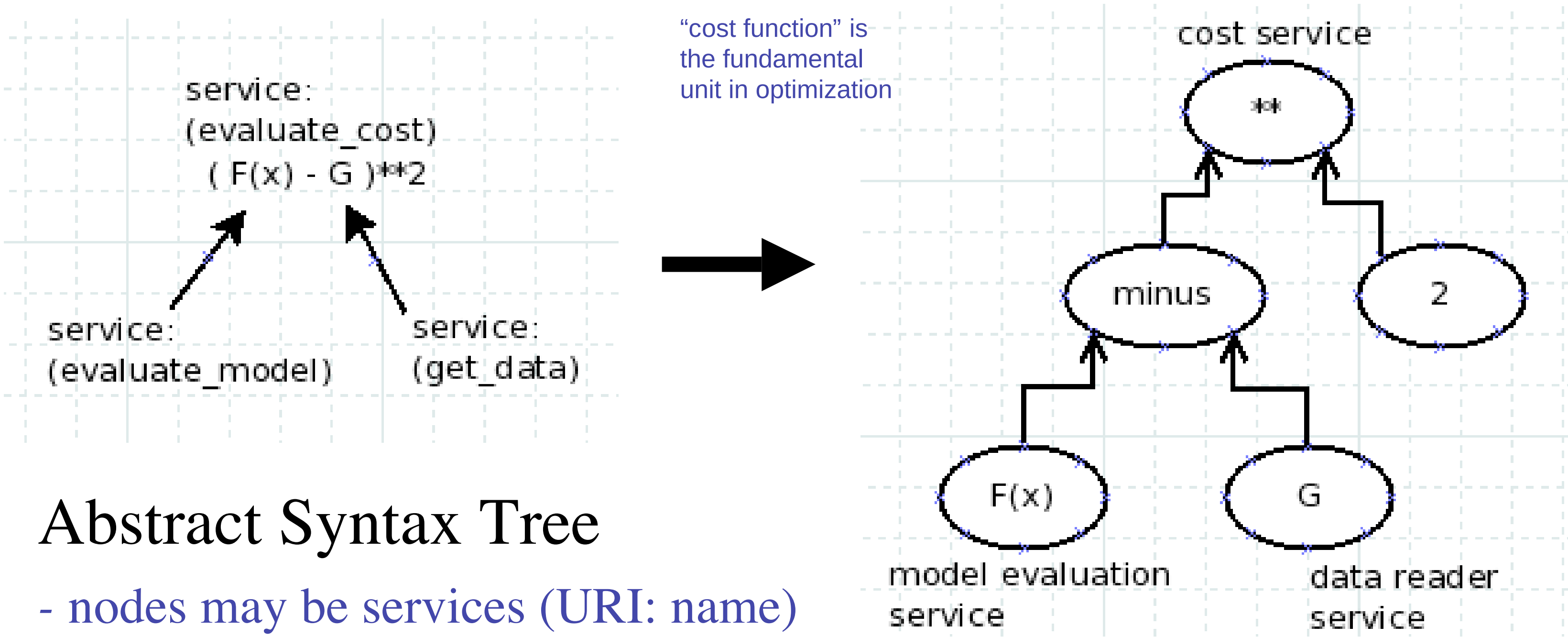


Mystic provides a stock set of configurable, controllable solvers with::

- a common interface
- the ability to impose solver-independent bounds constraints
- the ability to apply solver-independent monitors
- the ability to configure solver-independent termination conditions
- a control handler yielding: [pause, continue, exit, and user_callback]
- ease in selecting initial conditions: [initial_guess, random]

the cost function as an AST

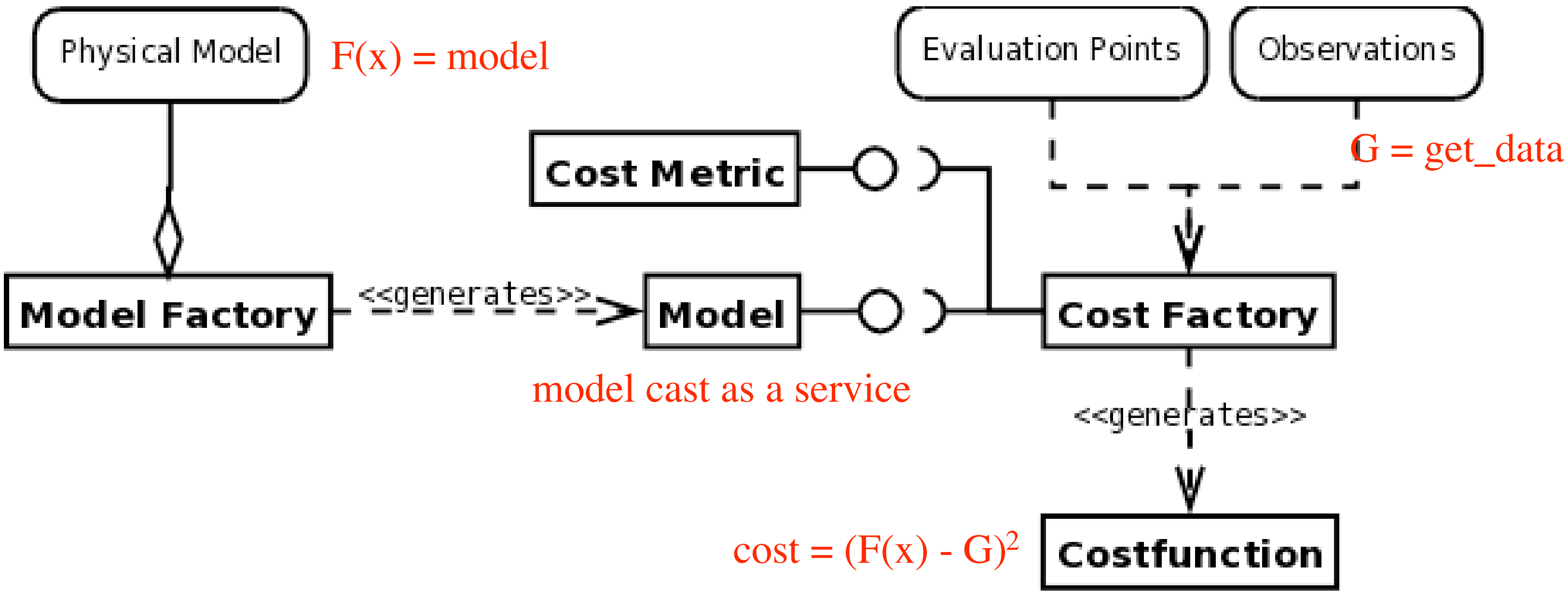
"cost function" is the fundamental unit in optimization



- Abstract Syntax Tree
  - nodes may be services (URI: name)
  - nodes may be simple operations
- Provides service infrastructure abstraction
  - service configuration and launching
  - service monitoring and management

building model & cost function

CALTECH PSAAP



- Factories hide the girdling of services from the user
  - model evaluation may require an entire computing cluster
  - data may not reside on the same machine as cost is evaluated
  - data may require some initial transformation

user's view of service-based models

CALTECH PSAAP

```
# "models" are class objects that act like functions:: result = function(evalpts)
# the 'factory' step can add infrastructure for coupling, constraints, derivatives, statistics, …
from mystic.models.lorentzian import Lorentzian
lorentz = Lorentzian(coeffs)   # a 'model factory'
y = lorentz(x)

# if the user provides a function, then a "model builder" should build a standard model
def identity(x):
  return x

from mystic.models import model_builder
my_model = model_builder(identity)
y = my_model(x)
```

configure & spawn a model instance

evaluate the model

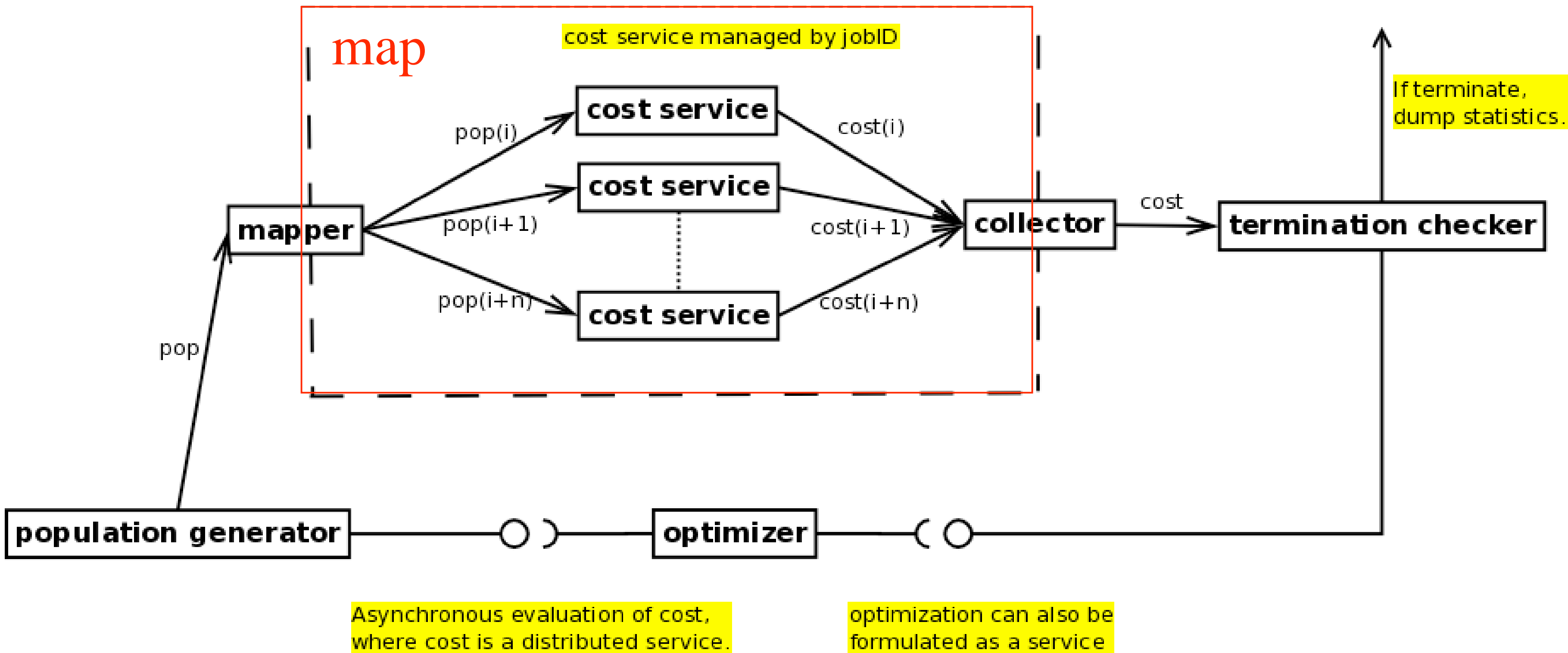
a generic model factory

- Configuration, launch, and evaluation are abstracted away
  - model object may be a proxy for a service located on another machine
  - model_builder provides a girdling mechanism for a user-defined function

optimization as a parallel service

CALTECH PSAAP

- ‘Map’ evaluates models in parallel
  - map hides complexity of underlying service connections
  - cost function cast as a distributed service (i.e. on another machine)
  - implies management of service communications (with job manager)



example: differential evolution

```
# the generalized solver algorithm
for generation in range(self._maxiter):

  StepMonitor(self.bestSolution[:], self.bestEnergy)

  …<generate a list of trial solutions trialPop> …

  trialEnergy = map(costfunction, trialPop)

  … <check if energy of trial solutions are lower than the current best energy> …

  if self._EARLYEXIT or termination(self):
    break

…
return


# here "map" is just python's map function
# however, it is a natural interface to lots of underlying complexity in job management
```

parallel differential evolution

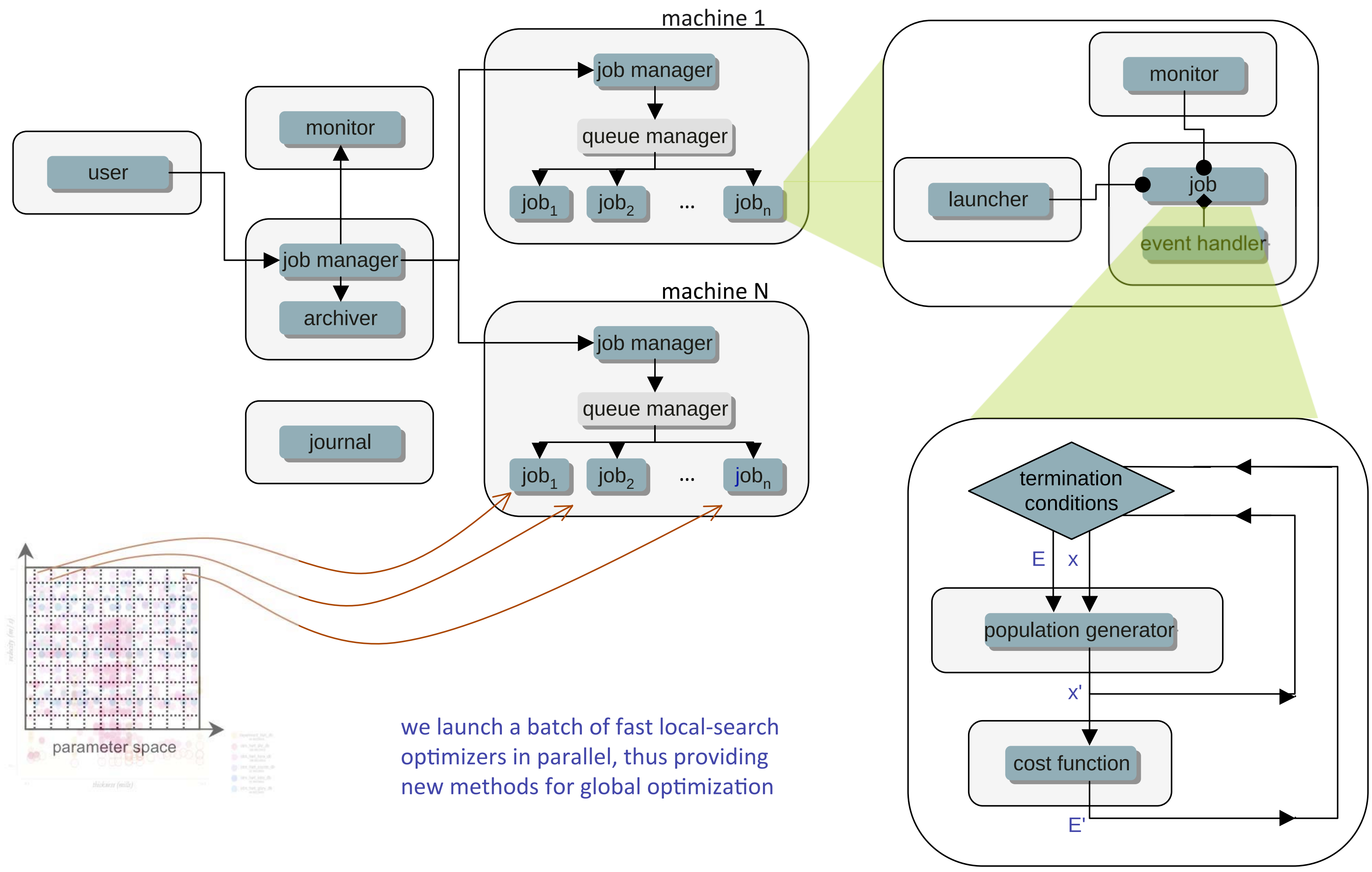
```
# the generalized solver algorithm
for generation in range(self._maxiter):

 # StepMonitor(self.bestSolution[:], self.bestEnergy)

 …<generate a list of trial solutions trialPop> …

 trialEnergy = map(costfunction, trialPop, processes=self._ncpus, servers=self._servers)

 … <check if energy of trial solutions are lower than the current best energy> …

 if self._EARLYEXIT or termination(self):
   break

…
return
```

```
trialEnergy = map(costfunction, trialPop, nnodes=self._nnodes,
                  launcher=self._launcher, mapper=self._mapper)
```

\# here “map” is the RPC-based parallel_map (or alternately, the MPI-based ez_map)
\# otherwise the solver code is identical

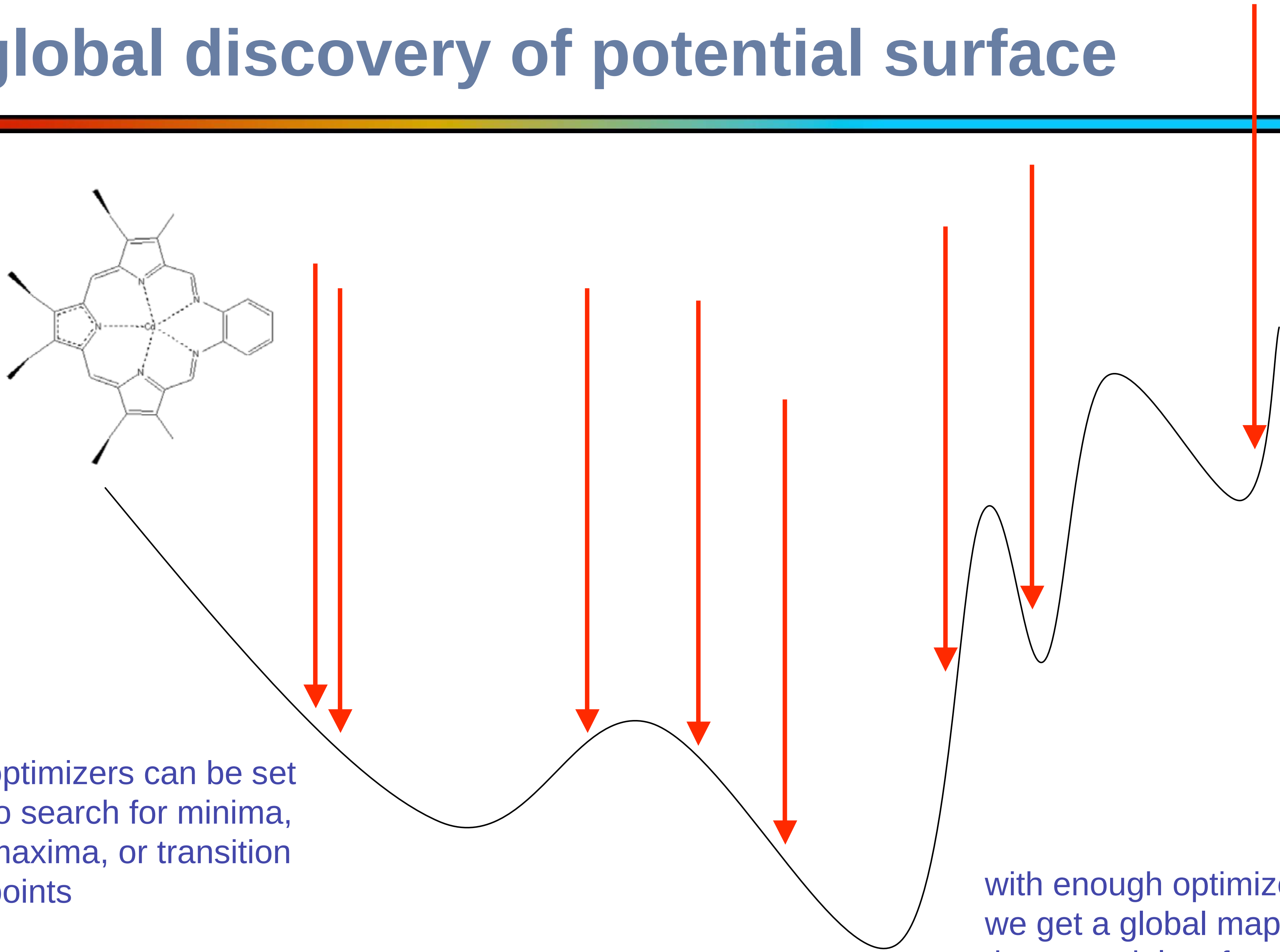
- …

fast exploration of parameter space



we launch a batch of fast local-search optimizers in parallel, thus providing new methods for global optimization

global discovery of potential surface



optimizers can be set to search for minima, maxima, or transition points

with enough optimizers, we get a global map of the potential surface

tools for building custom optimizers

monitorWrapper binds a monitor to an object

monitorWrapper

functionWrapper

functionWrapper binds a function to an object; useful for binding penalties to cost functions

costFactory generates a cost function from a set of parameters and N models

costFactory

penaltyFactory

penaltyFactory generates a penalty from symbolic and/or functional constraints

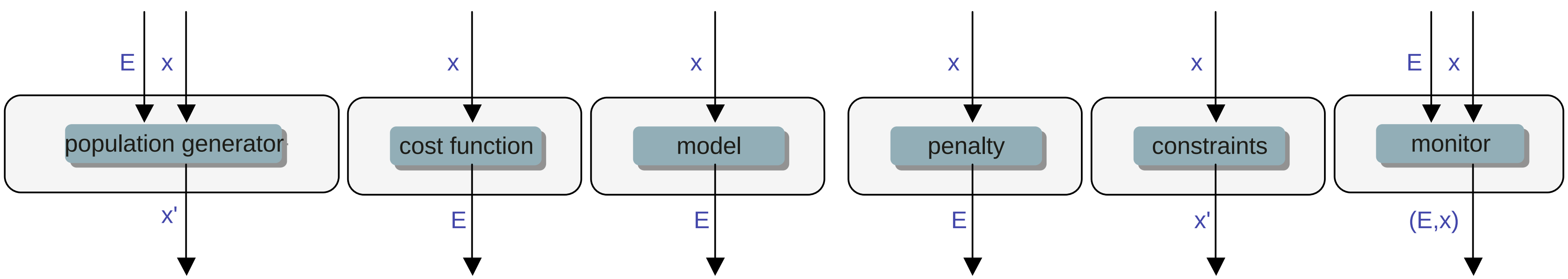
constraintsFactory generates constraints from symbolic and/or functional constraints

constraintsFactory

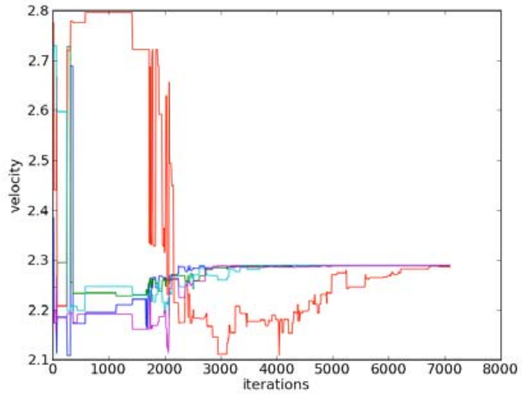
serviceFactory

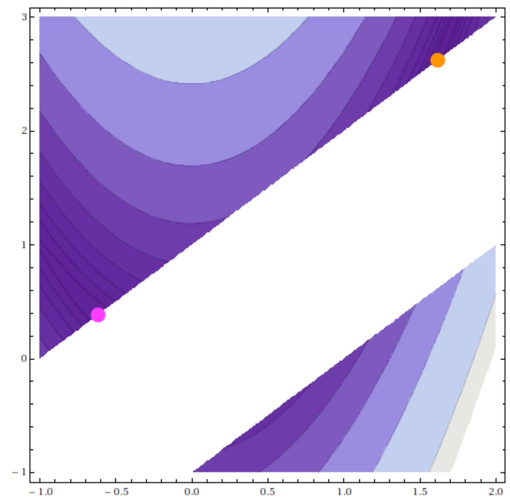
serviceFactory generates a compound service from a strategy and multiple copies of a service or services

- "special" models:
  - cost function = |model - model'|
  - penalty = ΔE if condition is False

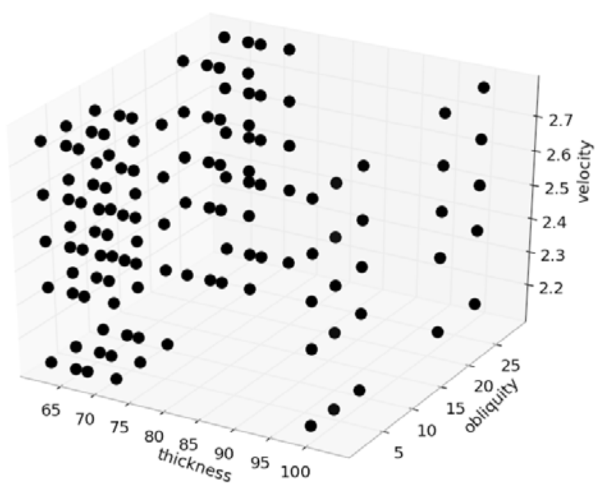


optimization analysis toolkit



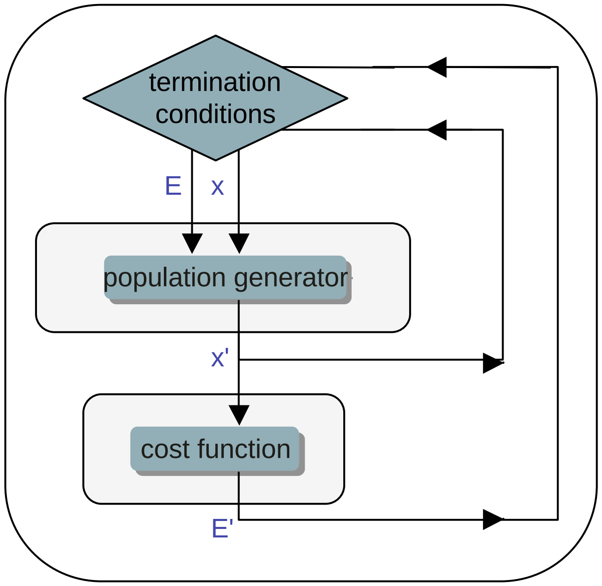


- monitoring and logging
  - parallel and distributed logging
  - generate convergence plots, contour plots, and parameter hypercube plots from logs
- workflow controls
  - dynamic stop, reconfigure, and restart capabilities for optimizers



optimization and difference metrics

- selection of optimizers
  - differential evolution
  - particle swarm
  - simulated annealing
  - branch and bound
  - nonlinear conjugate gradient
  - quasi-newton BFGS
  - Powell's directional search



an optimizer is composed of a population generator and termination conditions, acting on a cost function

a cost function provides a difference metric
- $E = |F(x) - G|^2$

- termination conditions are customizable across all optimizers

- rigorous sensitivity calculation as a global maximization of the cost function

- the diameter of a function measures the model variability over the range of input parameters
  - $E = |F(x) - F(x')|^2$ where $x_j = x'_j$ for $i \neq j$

measures of parameter sensitivity

CALTECH PSAAP

- $D_i[F] := sup\{| F(x) - F(y) | | x_j = y_j \; for \; i = j\}$

  - the diameter D of a function F measures the model variability over the range of given input parameters

  - diameter evaluations require the solution of a global optimization problem over the range of the inputs (define *optimal cost* $:= D_i[F]^2$)

- $D[F] := (D_1[F]^2 + \ldots + D_N[F]^2)^{1/2}$

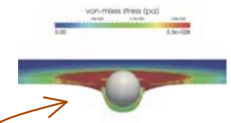
  - each independent variable has a sub-diameter D_i which can all be collected to provide a single measure of parameter impact on the model

- $D_i[F] \,/\, D[F]$

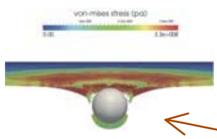
  - provides normalized measure of impact of a single parameter

diameter calculations

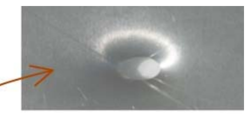
sensitivity



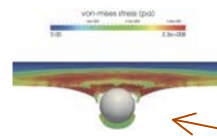
$|F(x) - F(x')|^2$



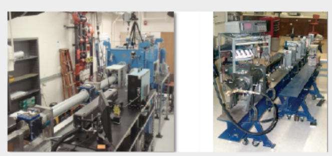
validation



$|F(x) - G(x)|^2$



the source of G(x) is a measurement from the SPHIR or HSRT facilities

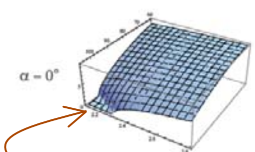


the source of $G_S(x)$ is a function call to an experiment surrogate equation

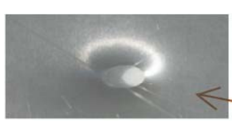
$$\text{ballistic limit: } v_{\text{bl}} := 0.5794 \left(\frac{h}{(\cos\alpha)^{0.4482}} \right)^{1.4004}$$

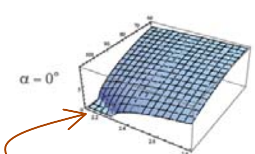
$$A_{\text{perf}} := \begin{cases} 0, & v < v_{\text{bl}}, \\ 10.3963 \left(\left(\frac{h}{1.778} \right)^{0.4757} (\cos\alpha)^{1.0275} \tanh\left(\frac{v}{v_{\text{bl}}} - 1 \right) \right)^{0.4682}, & v \geq v_{\text{bl}}. \end{cases}$$

Units: A_{perf} in mm^2, h in mm, α in radians, v in $\text{km} \cdot \text{s}^{-1}$.

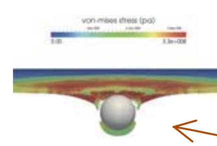


$|G(x) - G_s(x)|^2$





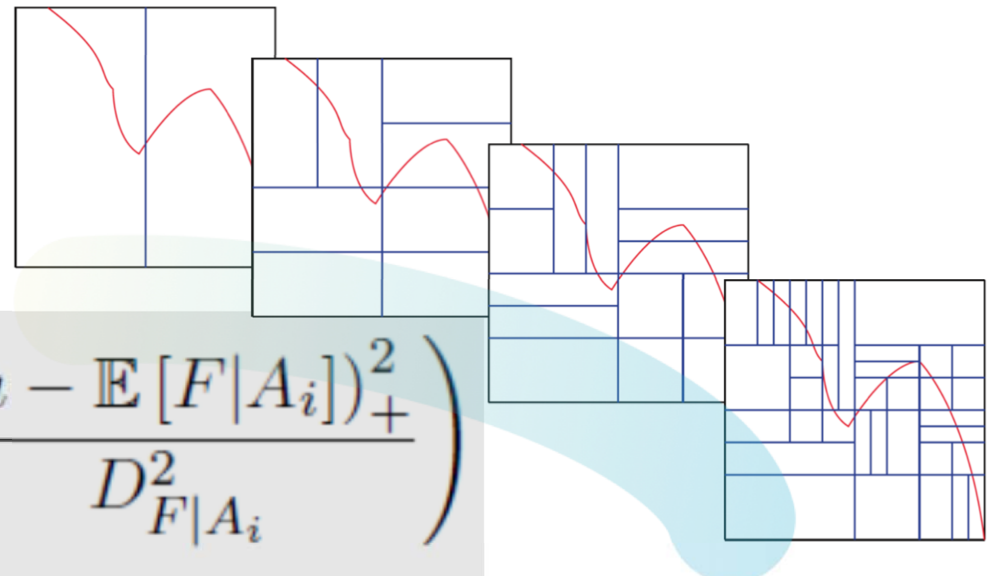
$|F(x) - G_s(x)|^2$



discovery of regions of criticality

CALTECH PSAAP

diameters help zero-in on regions of parameter space where parameters have desired impact



$$\sum_{i=1}^{N} \text{Prob}\left[A_i\right] \exp\left(-2\frac{\left(a - \mathbb{E}\left[F|A_i\right]\right)_+^2}{D_{F|A_i}^2}\right)$$

- Partitioning to find ‘interesting’ regions of parameter space
  - calculate PoF upper bound for each region of parameter space
  - identify regions where PoF = 1 or 0; remove as ‘uninteresting’
  - select region with highest contribution to the PoF upper bound
  - divide parameter space along the axis with largest diameter

probability & statistics toolkit

- probability and statistical analysis tools
  - McDiarmid-based UQ calculators
  - parallel Monte Carlo calculators
  - parameter sensitivity and model validation
  - optimization over probability measures
  - basic statistics and probability functions

$$\phi(\vec{x}) = f(c(\vec{x}))$$

"direct" application of a constraints function

- standard functions like mean and range are provided
  - not only ability to measure the quantity, but to impose it
  - statistics can be imposed on a set algebraically or numerically (using an optimizer)
  - implemented to conserve of other properties when possible
  - facilitate decoupling constraints from the optimization problem

constraints toolkit

$$\phi(\vec{x}) = f\left(c(\vec{x})\right)$$

set-based constraints are applied as a change on the x-axis and thus reduce the set to valid parameter space

$$\phi(\vec{x}) = f(\vec{x}) + k \cdot p(\vec{x})$$

penalty-based constraints are applied as a change on the y-axis

- linear, nonlinear, and numerical solvers used to simplify constraints
- constraints toolkit reuses existing mechanisms
  - constraints applied with model coupler
  - constraints solved or iteratively enforced by nested optimization

- several penalty methods available
  - barrier and penalty methods
  - augmented Lagrange multiplier method
  - customized and compound methods

- decoupling constraints enables solving of highly-constrained optimization problems

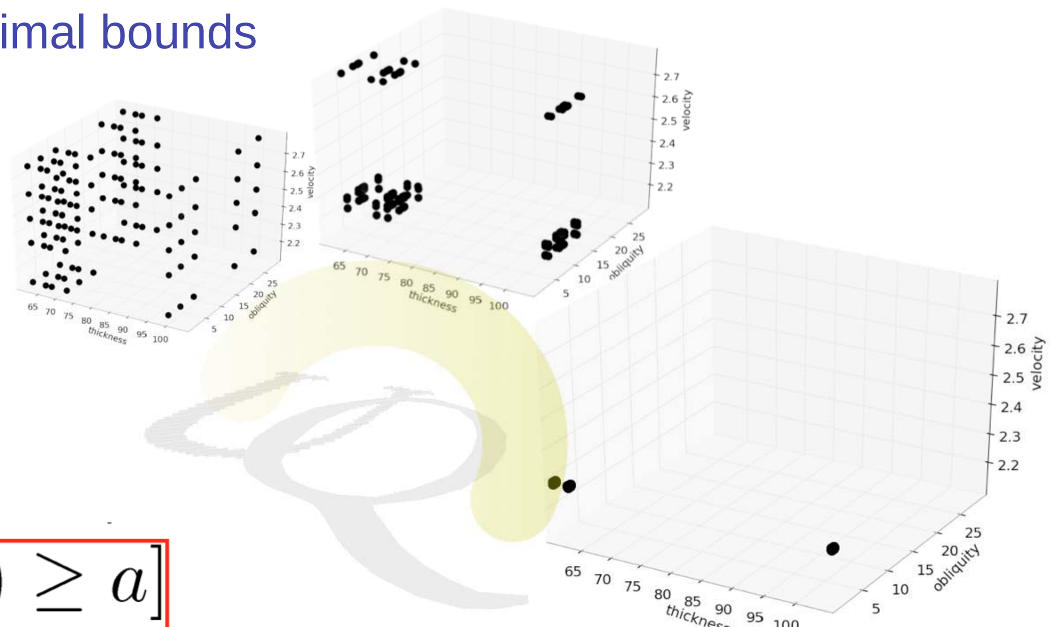
optimal uncertainty quantification

- optimal uncertainty quantification (OUQ) is maximization over a probability distribution, not over a cost function
  - distribution is determined by the constraints applied to a probability measure
  - determines the optimal bounds

$$\mathcal{A}_1 = \left\{ \begin{array}{l} \{\mu, H\} : \\ H : \chi \to \mathbb{R} \\ \mu \in \mathcal{P}_1(\chi) \\ \mathbb{E}_\mu[H] = m \\ (\max H - \min H) \leq D \end{array} \right\}$$

$$p_{\max}\delta_a + (1 - p_{\max})\delta_{a-D}$$

the optimizer tracks the collapse of Dirac masses



(1) $$\sup_{(\mu,H)\in\mathcal{A}} \mu[H(X) \geq a]$$

Claim: (1) Is the optimal/best/sharpest upper bound on the POF

$$\mathcal{U}(\mathcal{A}_1) = p_{\max} := \left(1 - \frac{(m-a)_+}{D}\right)_+$$

measures as data objects

By selecting Diracs as the basis of the 1D measures, the optimizer can discover the weights and positions of the Diracs that maximize $\mu[H = 0]$. For support from any independent random variable x, we can write:

$$\mu_x = \sum_{i=1}^{N_x} w_{x_i} \delta_{x_i}; \text{ where } 1 = \sum_{i=1}^{N_x} w_{x_i}. \tag{6.4}$$

We can also express $\mu[H = 0]$ and $E_\mu[H]$ in terms of Diracs on a 3D product measure:

$$\mu[H = 0] = \sum_{i,j,k} w_{x_i} w_{y_j} w_{z_k} \mathbb{1}[H(x_i, y_j, z_k) = 0] \tag{6.5}$$

$$E_\mu[H] = \sum_{i,j,k} w_{x_i} w_{y_j} w_{z_k} H(x_i, y_j, z_k) \tag{6.6}$$

information alters probability of failure

Admissible scenarios, $\mathcal{A}$	$\mathcal{U}(\mathcal{A})$	Remarks
$\mathcal{A}$: independence and mean constraints	$\leq 66.4\%$ $= 43.7\%$ $= 37.9\%$	McDiarmid's inequality Optimal McDiarmid *mystic*, H known
$\mathcal{A} \cap \left\{ \mu \,\middle\|\, \begin{matrix} \mu\text{-median velocity} \\ = 2.45\,\mathrm{km}\cdot\mathrm{s}^{-1} \end{matrix} \right\}$	$= 30.0\%$	*mystic*, H known
$\mathcal{A} \cap \left\{ \mu \,\middle\|\, \mu\text{-median obliquity} = \frac{\pi}{12} \right\}$	$= 36.5\%$	*mystic*, H known
$\mathcal{A} \cap \left\{ \mu \,\middle\|\, \text{obliquity} = \frac{\pi}{6}\ \mu\text{-a.s.} \right\}$	$= 28.0\%$	*mystic*, H known
$\mathcal{A}'' := \{\text{uniform measure}\}$	$= 3.8\%$	Exact calculation

We keep trying possible "experiments" to find the information set that certifies the system as "safe" (not failing within tolerance)

We can apply this design of experiments to materials discovery?

rigorous probability of failure for terminal ballistics

M. Ortiz et al.

CALTECH PSAAP

FEATURED COLLABORATIONS

RESEARCH OBJECTIVES

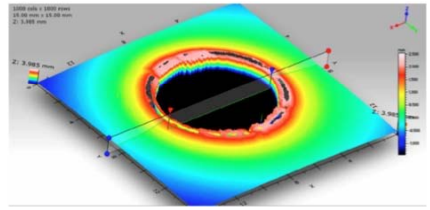
To devise a methodology for uncertainty quantification that leads to tight upper bounds on the probability of failure for complex systems.

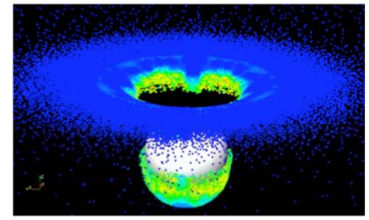
APPROACH

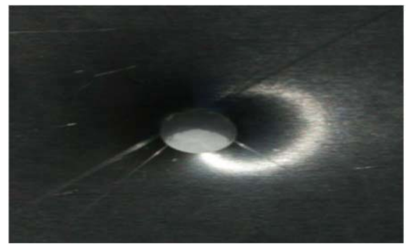
Systems where uncertainty in the response of the system is aleatoric (assumed to stem from randomness of the system inputs) are ideal to be described with McDiarmid's inequality as a basis for rigorous uncertainty quantification. The terminal ballistics of aluminum plates impacted by spherical steel projectiles provides such a system, where plate penetration serves as the failure criterion.

The oscillation of the response function for ballistic impact is calculated as a global optimization problem using ***mystic***. The staging and launching of thousands of optimal-transportation models (OTM) of ballistic penetration is automated with ***pathos***.

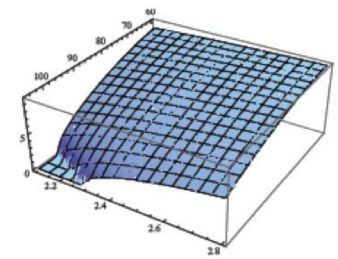
otm CALTECH PSAAP







$\alpha = 0°$



Several instances of ballistic penetration used in the certification of Caltech's gas-gun facility. Clockwise from bottom left are a snapshot of experimental results, an optical scan of the penetrated plate, a simulation of plate penetration, and a plot of an analytical surrogate model derived from experimental results.

SIGNIFICANT RESULTS

A paper describing the rigorous uncertainty quantification analysis used to certify the lethality of a steel projectile impacting an aluminum plate is forthcoming.

BROADER IMPACT

The methodology described in the paper is broadly applicable to complex systems with aleatoric uncertainty.

uses Pathos
uses Mystic
uses UQ

real-time analysis for parametric structure refinements

S. Billinge et al.

CALTECH PSAAP

RESEARCH OBJECTIVES

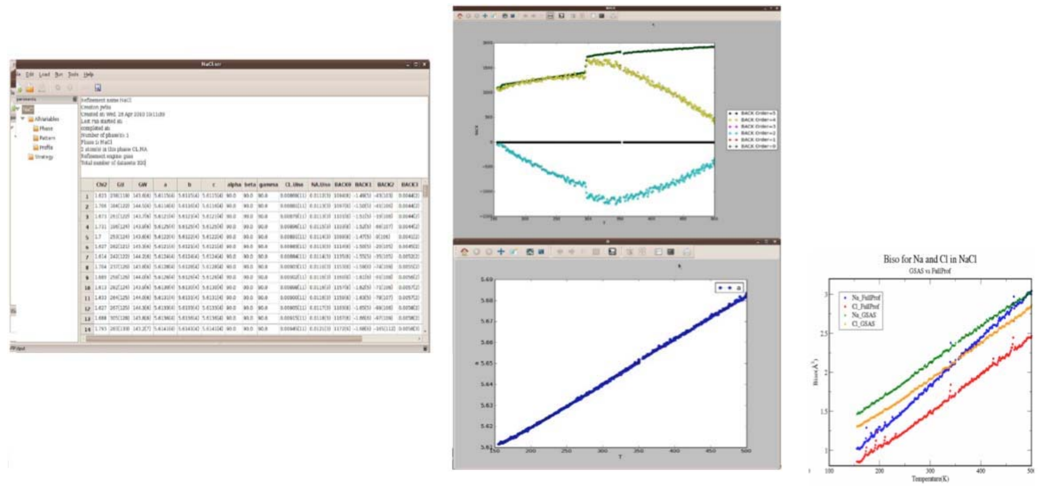
To develop software to overcome the bottleneck of automating the extraction of scientific results from the terabytes of raw data produced by diffractometers at the SNS.

APPROACH

SrRietveld is innovative structure refinement software that provides a highly-automated and easy-to-use interface for two of the most popular *Rietveld* refinement programs. *SrRietveld* provides the scripting layer that executes and manages the individual refinement engines. Parametric studies are crucial for understanding materials systems -- *SrRietveld* uses ***pathos*** to launch parametric refinements in parallel on heterogeneous resources.

SrRietveld

DANSE



Screenshots of SrRietveld executing over 300 refinements of NaCl in parallel. SrRietveld can be used to study the effect of temperature, pressure, and chemical composition on materials parameters. When refinements are run in parallel on a computing cluster, SrRietveld enables real-time analysis of data for the high-throughput diffractometers at the SNS.

SIGNIFICANT RESULTS

SrRietveld is designed to provide real-time analysis for parametric data, and has been deployed on high-throughput diffractometers at the SNS. A paper on *SrRietveld* is forthcoming.

BROADER IMPACT

SrRietveld is open source software. An upcoming releasecapable of executing massively parallel batch parametric refinements will be available to the broader scientific community at http://www.diffpy.org/doc/srrietveld.

uses Pathos

FEATURED COLLABORATIONS

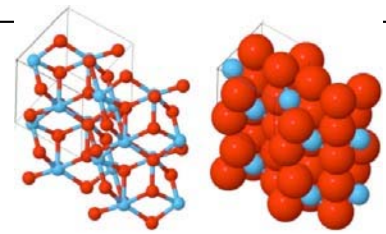
sensitivity of phonon energy to interatomic bonding

C. Li, M. McKerns , B. Fultz

CALTECH PSAAP

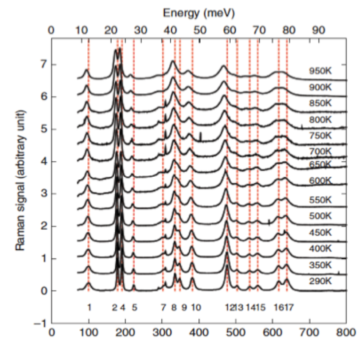
RESEARCH OBJECTIVES

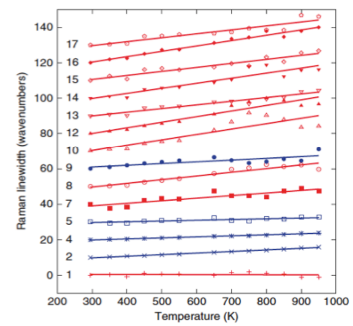
To determine the nature of phonon anharmonicity in zirconia at elevated temperatures.

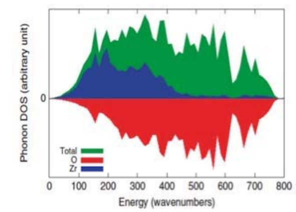


APPROACH

Raman spectra of zirconia were measured at temperatures up to 950K. Raman peak shifts and broadenings with increasing temperature were analyzed for trends. Lattice dynamics calculations were performed with GULP, using a shell model to obtain Raman frequencies and densities of states. ***Mystic*** is used to calculate the sensitivity of the shell model to the selected force field parameters, with change in the phonon energy serving as the performance measure. Correlations between significant terms in the shell model and atomic motions calculated by GULP were then noted.







Crystal structure of monoclinic zirconia, with oxygen (O) in red and zirconium (Zr) in blue. The partial density of states at 295K calculated with GULP shows Zr dominates the lower energy modes. Experimental data shows metal-dominated modes (blue) broaden minimally.

SIGNIFICANT RESULTS

By correlating atomic motions to thermal peak shifts and broadenings, modes involving changes to oxygen-oxygen bond lengths were determined to be the most anharmonic. Metal-dominated modes were found to be more quasiharmonic, and thus broaden less with temperature. *Published: C. Li et al., JACerS, 2010.*

BROADER IMPACT

This study yields a methodology for elucidating the nature of phonon anharmonicity in bulk metal oxides at elevated temperatures.

uses Mystic

uses UQ

DANSE

FEATURED COLLABORATIONS

validation of materials strength models of ballistic impact

D. Meiron et al.

CALTECH PSAAP

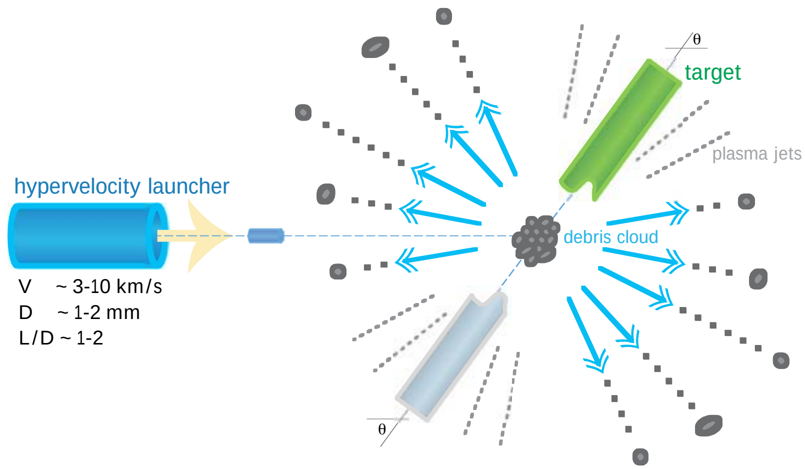
RESEARCH OBJECTIVES

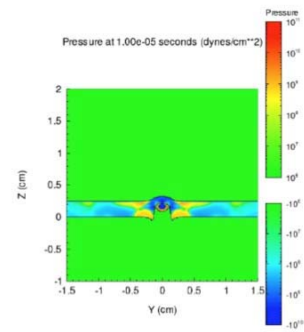
To validate materials strength models of hypervelocity ballistic impact.

APPROACH

Eulerian hydrocodes, such as CTH, provide several models for materials deformation. Three popular materials strength models (Von Mises yield surface, Johnson-Cook viscoplastic, and Steinberg-Guinan-Lund constitutive) were selected for validation against experimental data. ***Pathos*** is used to stage and launch thousands of materials strength models of spherical steel projectiles impacting steel plates. ***Mystic*** is used to calculate the sensitivity of the strength models to selected input parameters, with plate penetration serving as the performance measure.

CTH + CALTECH PSAAP





Schematic of hypervelocity impact for a spherical steel projectile fired at stainless steel plates. Over fifty experimental shots with different impact velocities, impact angles, and plate thicknesses were performed on Caltech's SPHIR facility. A Von Mises yield strength model with velocity = 1000 m/s is shown at 1e-5 s after impact.

SIGNIFICANT RESULTS

Thus far, for a steel projectile impacting a stainless steel plate, the Von Mises model produces an accurate perforation area but a low ballistic limit, while the Johnson-Cook model produces an accurate ballistic limit but too a large perforation area. Sensitivity analysis is underway.

BROADER IMPACT

A result of this study will be to develop a methodology for improving materials strength models of hypervelocity impact.

uses Pathos
uses Mystic
uses UQ

FEATURED COLLABORATIONS

PSAAP: Predictive Science Academic Alliance Program

validation of a force field model for hypervelocity impact

W. Goddard et al.

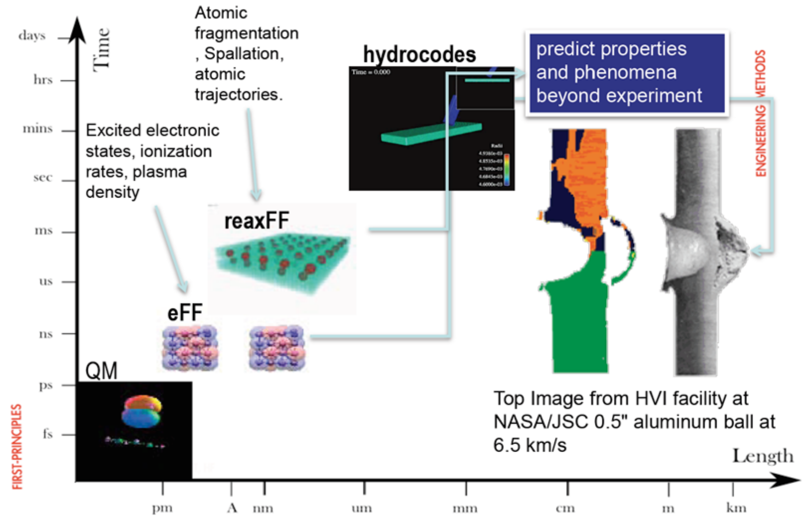
CALTECH PSAAP

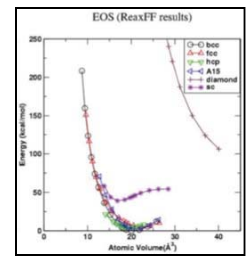
RESEARCH OBJECTIVES

To develop a force field model of the reactive processes induced in bulk tantalum under shock or thermal impact.

APPROACH

ReaxFF provides a first-principles-based description of complex reactive processes involving millions of atoms. *ReaxFF* uses force fields with parameters optimized to reproduce quantum mechanical (QM) results on relevant condensed phase structures and relevant quantities such as surface energy and stacking faults energy. Calculated materials properties are validated against equation of state experimental data and QM calculated results. In order to better identify the critical force field parameters, ***mystic*** is used to calculate the sensitivity of the *ReaxFF* model to selected force field parameters, with change in the bulk modulus serving as the performance measure. The staging and launching of thousands of *ReaxFF* models of bulk tantalum is executed using ***pathos***.





SIGNIFICANT RESULTS

Thus far, a *ReaxFF* model has been developed for bulk tantalum, however force field parameters need tuning to correct fragment trajectories. Sensitivity analysis is underway.

BROADER IMPACT

A result of this study will be to produce a force field model of hypervelocity impact for tantalum.

ReaxFF and eFF provide first-principles-based initial conditions for continuum plasma simulations. ReaxFF calculations of atomic packing of tantalum were verified against QM calculations with SeqQuest.

uses Pathos
uses Mystic
uses UQ

ReaxFF

FEATURED COLLABORATIONS

FURTHER COLLABORATIONS

- To develop a framework for rigorous ***uncertainty quantification*** utilizing *legacy data.*

  H. Owhadi et al.

- To employ sensitivity-based partitioning of parameter space to *discover* the key *enzymatic reactions* in synaptic plasticity.

  M. Kennedy et al.

- To develop massively parallel optimization algorithms to enable the efficient *mapping* of enzyme *reaction surfaces.*

  W. Goddard et al.

- To *extract* molecular *structure* of *colloidal materials* from associated small angle scattering data.

  P. Butler et al.

- To certify the *safety* of a structure under *seismic ground motion*, knowing only magnitude and focal distance of a nearby earthquake.

  M. Ortiz et al.

- To discover models for the *distribution* of *microstructures* that provide optimal multiphase *alloy materials response.*

  M. Ortiz et al.

PYRE FRAMEWORK CONTACT
MICHAEL AIVAZIS / AIVAZIS@CALTECH.EDU

MYSTIC/PATHOS FRAMEWORKS CONTACT
MIKE MCKERNS / MMCKERNS@CALTECH.EDU

CACR CONTACT
MARK STALZER / INFO@CACR.CALTECH.EDU

CONTACTS

End Presentation