GLOBAL DETERMINISTIC AND STOCHASTIC OPTIMIZATION IN A SERVICE ORIENTED ARCHITECTURE

Chaitra Raghunath[a], **Tyler H. Chang**[a], **Layne T. Watson**[abc]
Mohamed Jrad[c], **Rakesh K. Kapania**[c]
Departments of Computer Science[a],
Mathematics[b], and Aerospace & Ocean Engineering[c]
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0106 USA

Raymond M. Kolonay
AFRL/RQVC
2210 8th Street, Bldg. 146
Wright-Patterson Air Force Base
Dayton, OH 45433

http://people.cs.vt.edu/~thchang/SORCER.pdf

Virginia Tech

Multidisciplinary Design Optimization (MDO)

Consider the MDO of an aircraft design problem:

- Used during design space exploration (conceptual design step)
- Goal of achieving optimal design over multiple disciplines
- Reduces size of potential design space in future steps

Problem: Traditional MDO uses low fidelity models with poor accuracy

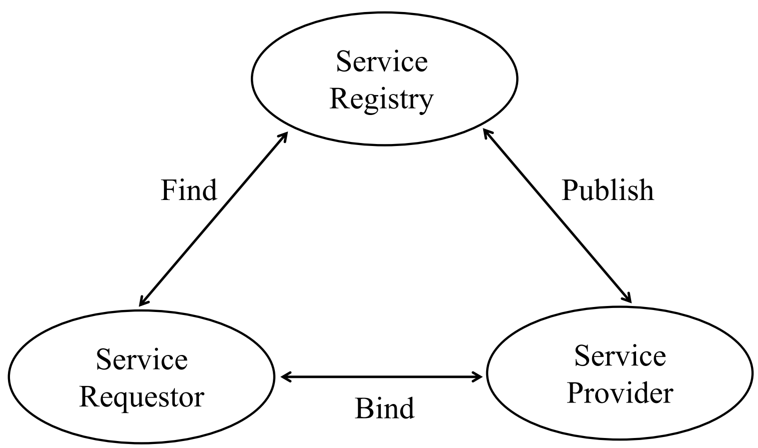
Potential Solution: Higher fidelity physics-based modeling tools

Drawback: High fidelity models can be prohibitively complex

Service Oriented Architectures & SORCER

Service Oriented Architecture (SOA) provides a framework for distributed computing:

- Homo- and/or heterogeneous resources are interoperable, reusable, and loosely coupled services
- Dynamically allocate resources upon service request
- Service ORiented Computing EnviRonment (SORCER) layered over FIPER metacompute grid



Optimization Algorithms

In this work we consider two optimization algorithms used in MDO:

VTDirect95

- For deterministic global optimization
- Fortran 95 implementations of D. R. Jones' Dividing Rectangles (DIRECT) algorithm
- Parallel and serial codes

QNSTOP

- For stochastic global optimization
- Quasi-Newton method in Fortran 2003
- Parallel and serial codes

Objectives

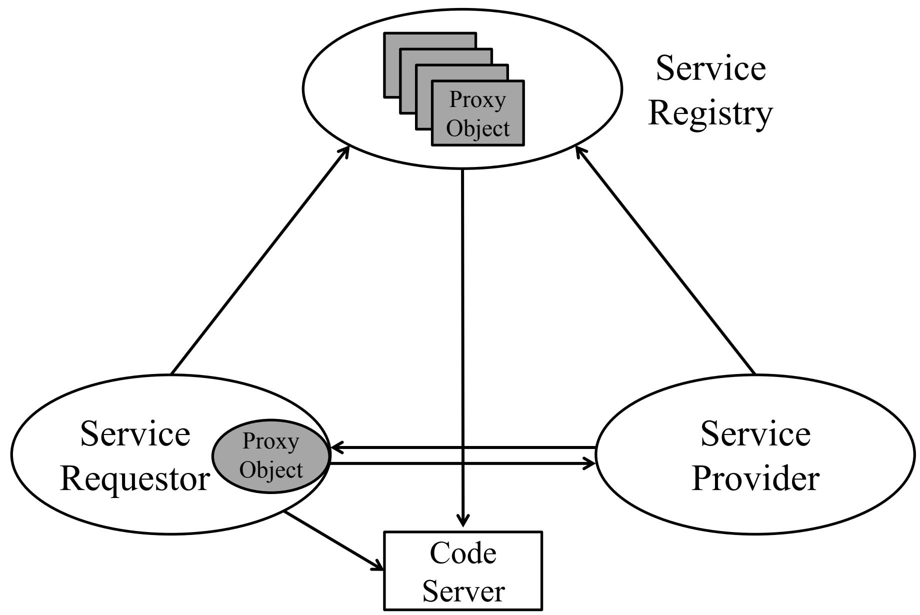
Provide VTDirect95 and QNSTOP as services on a SORCER grid

- Dynamically distribute function evaluations

Study the overhead of using SORCER for distributed optimization

Background: SORCER - SOOA

Service **Object**-Oriented Architecture (SOOA)



Background: SORCER - Implementation

Implementation

- Uses Jini (Apache River) connection technology
- Java based services (for interoperability)
- Leverages JavaSpaces for dynamically load balanced network

Service provider types:

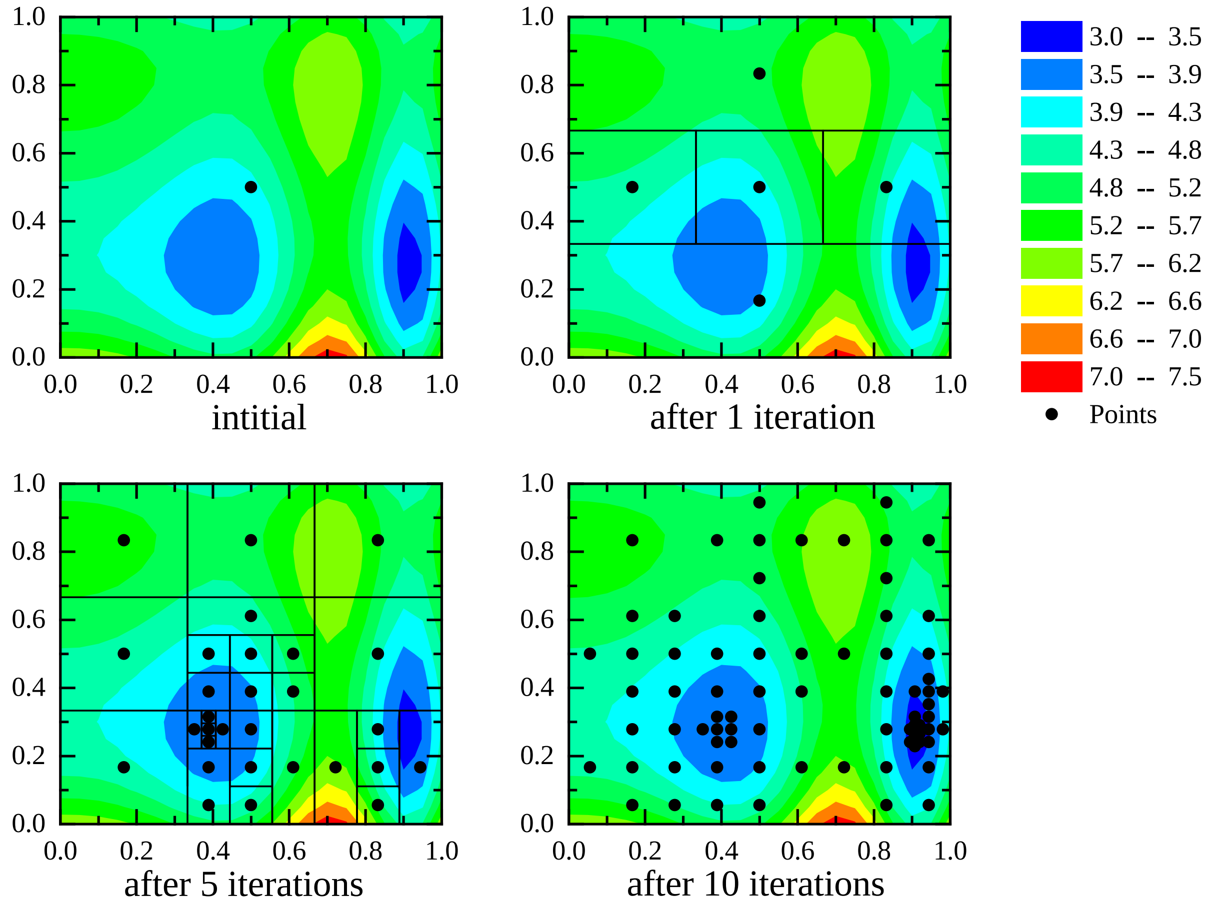
- Analysis providers
- Model providers

Abstraction layers

- Exertion-oriented programming (EOP)
- Var-oriented programming (VOP)
- Var-oriented modelling (VOM)

Background: VTDIRECT95 - basic algorithm

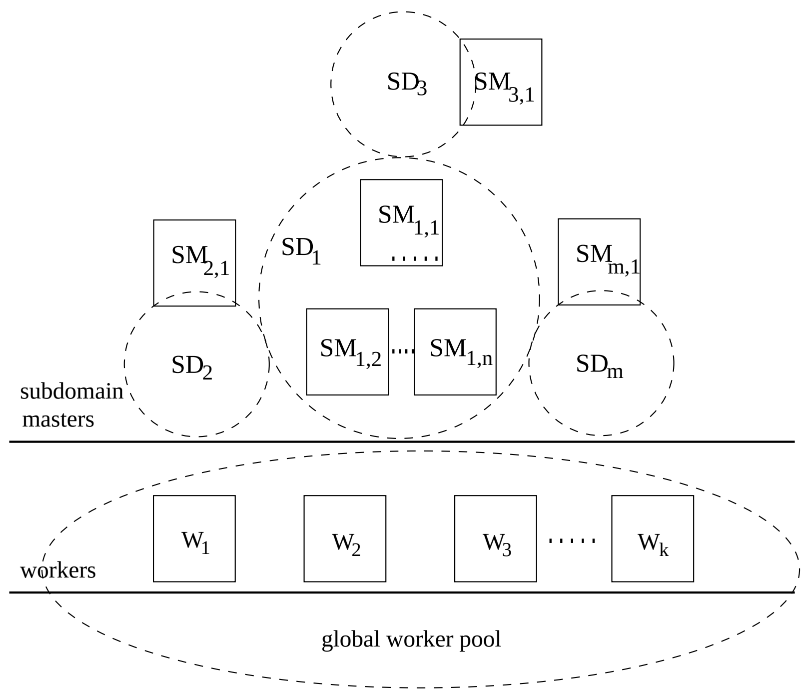
Based on Dividing Rectangles (DIRECT) by D.R. Jones



Background: VTDIRECT95 - parallel algorithm

The parallel VTDIRECT95 algorithm (pVTdirect) is fully distributed:

- Problem divided between multiple masters to share memory burden
- Function evaluation tasks distributed to workers



Background: QNSTOP - algorithm

Step 1 (regression experiment): Given a feasible set Θ, a current iterate X_k, and a radius τ_k:

- Compute the ellipsoidal design region $E_k(\tau_k)$ centered at X_k
- Compute LS estimate for the gradient $\hat{g}_k$ from uniform sampling of $E_k(\tau_k)$

Step 2 (secant update): Estimate Hessian matrix $\hat{H}_k$.

Step 3 (update iterate): Calculate the next iterate X_{k+1} from a scaling matrix W_k and lagrange multiplier μ_k

- Project X_k onto the feasible set Θ

Step 4 (update subsequent design ellipsoid): Compute an updated scaling matrix W_{k+1}.

Step 5: If room for more function evaluations in budget go to **Step 1**. Otherwise, the algorithm terminates.

Background: QNSTOPP - parallelism

Parallel Algorithm **QNSTOPP** (w/ OpenMP)

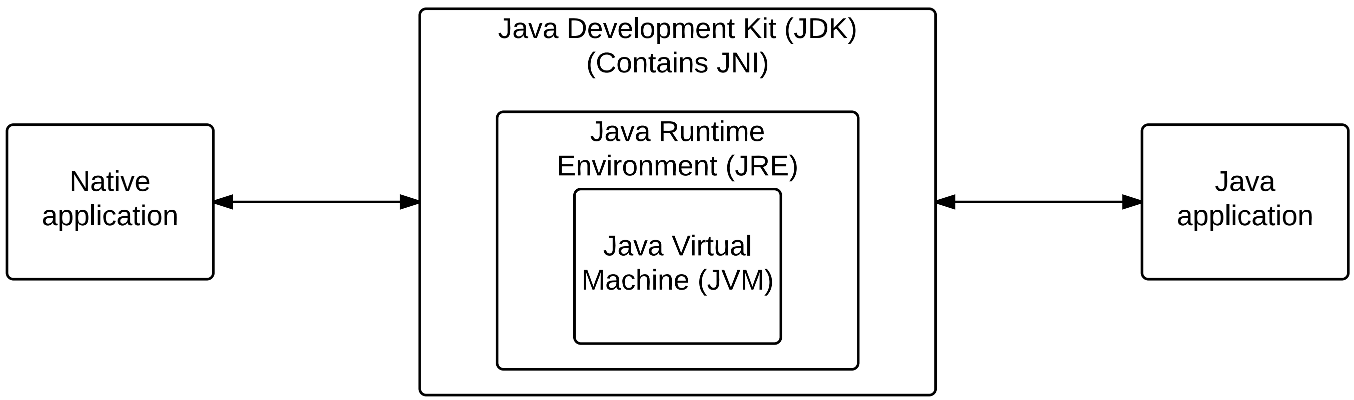
Sources of parallelism:

- Individual function evaluations
- Loop over samples in experimental design
- Loop over start points

Method: JNI Wrappers

Java Native Interface (JNI) libraries used to wrap Fortran optimization code in Java (as SORCER *analysis* service)

- Leverage *invocation interface* to allow native C/C++ code to run in JVM
- C “glue code" needed to wrap Fortran routines
- Objective functions are analysis providers invoked by optimization algorithm



Method: pVTdirect

Parallel VTDIRECT95 subroutine (pVTdirect) fundamentally incompatible with SORCER

- SORCER assumes master/slave paradigm
- pVTdirect is fully distributed for scalability

Method: QNSTOPP

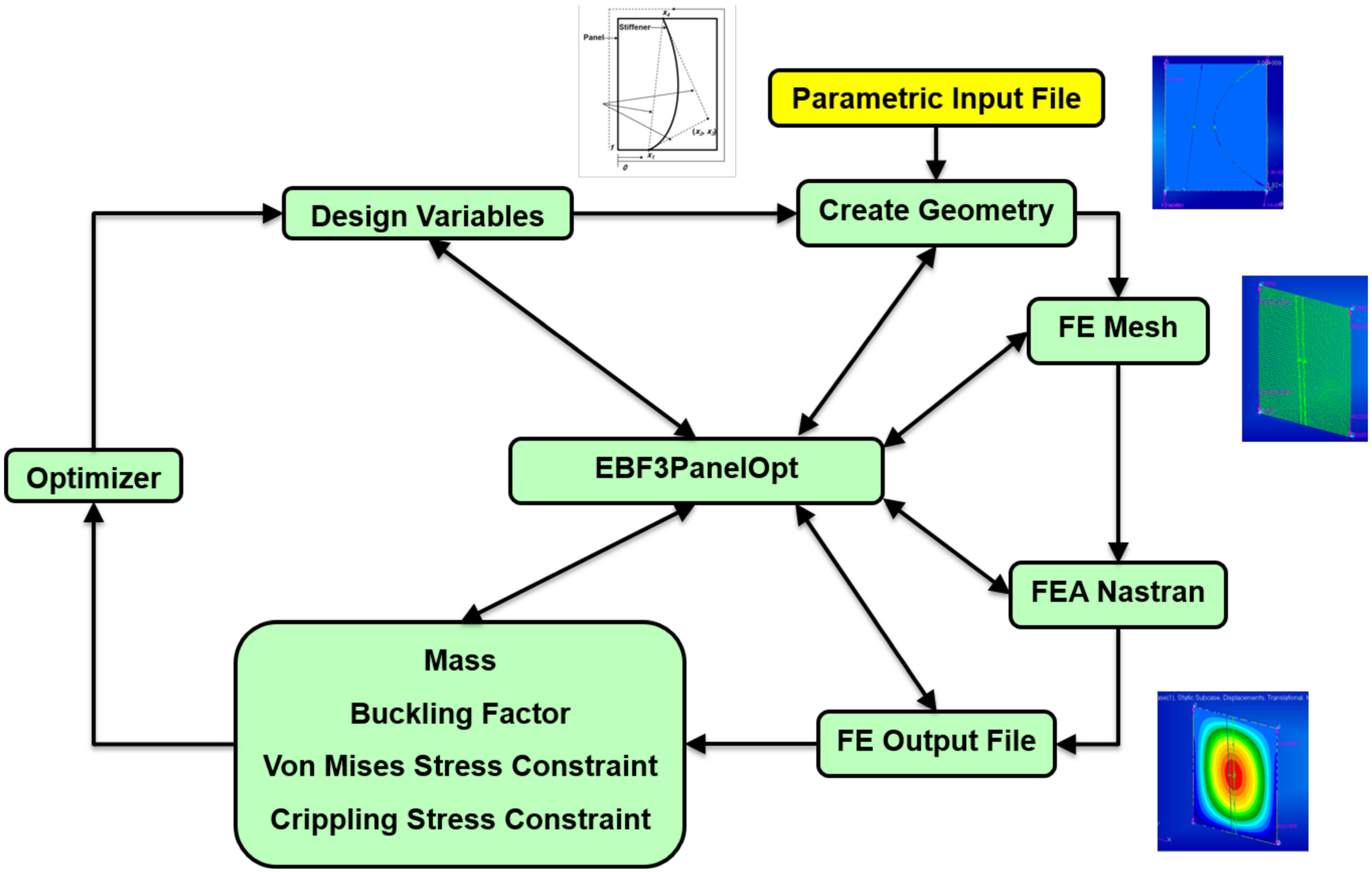
Parallel QNSTOP subroutine (QNSTOPP) parallelized over sampling of design region

- Chunked out so that n function evaluations are requested at a time via SORCER service requests
- Leads to n way parallelism wrt objective function evaluations

Experiment: EBF3PanelOpt

Framework for optimization of curvilinearly stiffened panels (wrt panel mass)

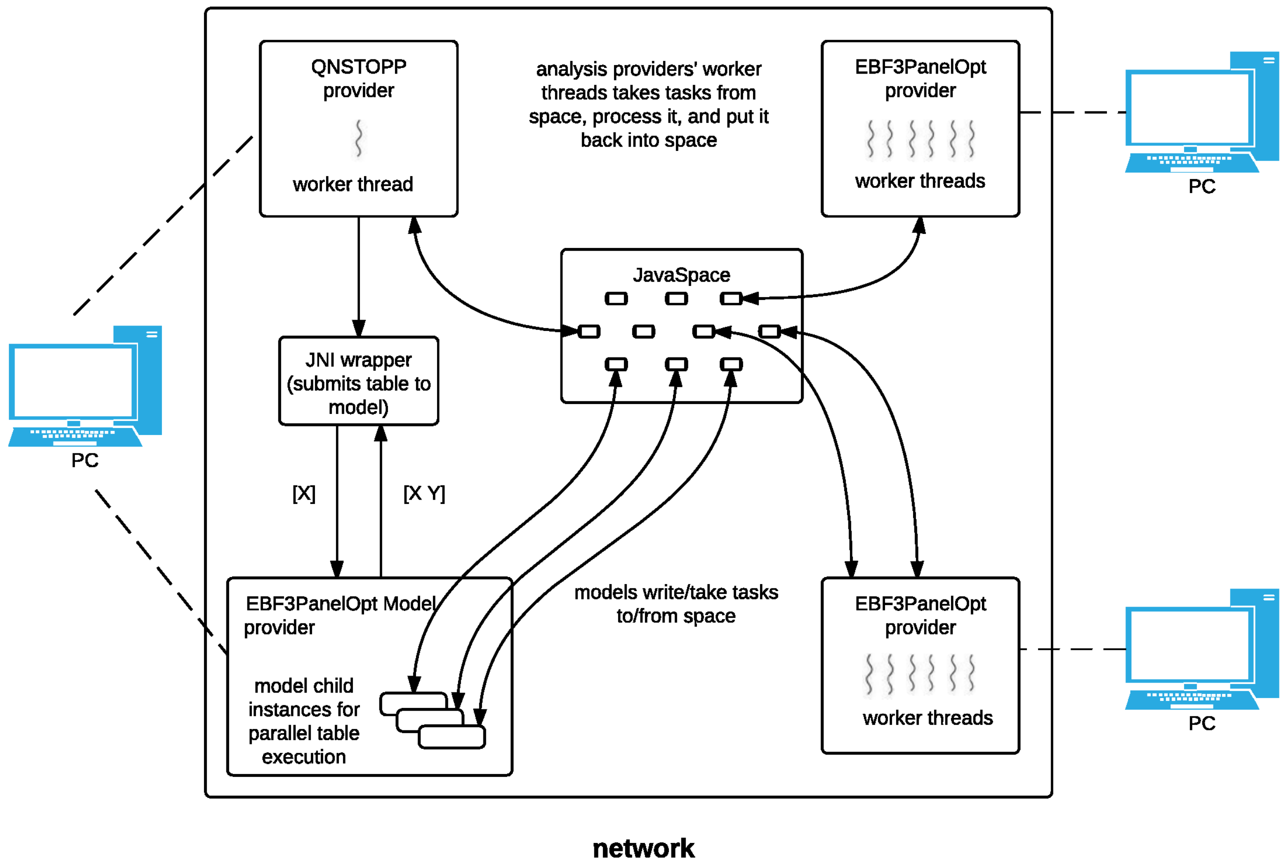
- Python based



Experiment: EBF3PanelOpt Implementation

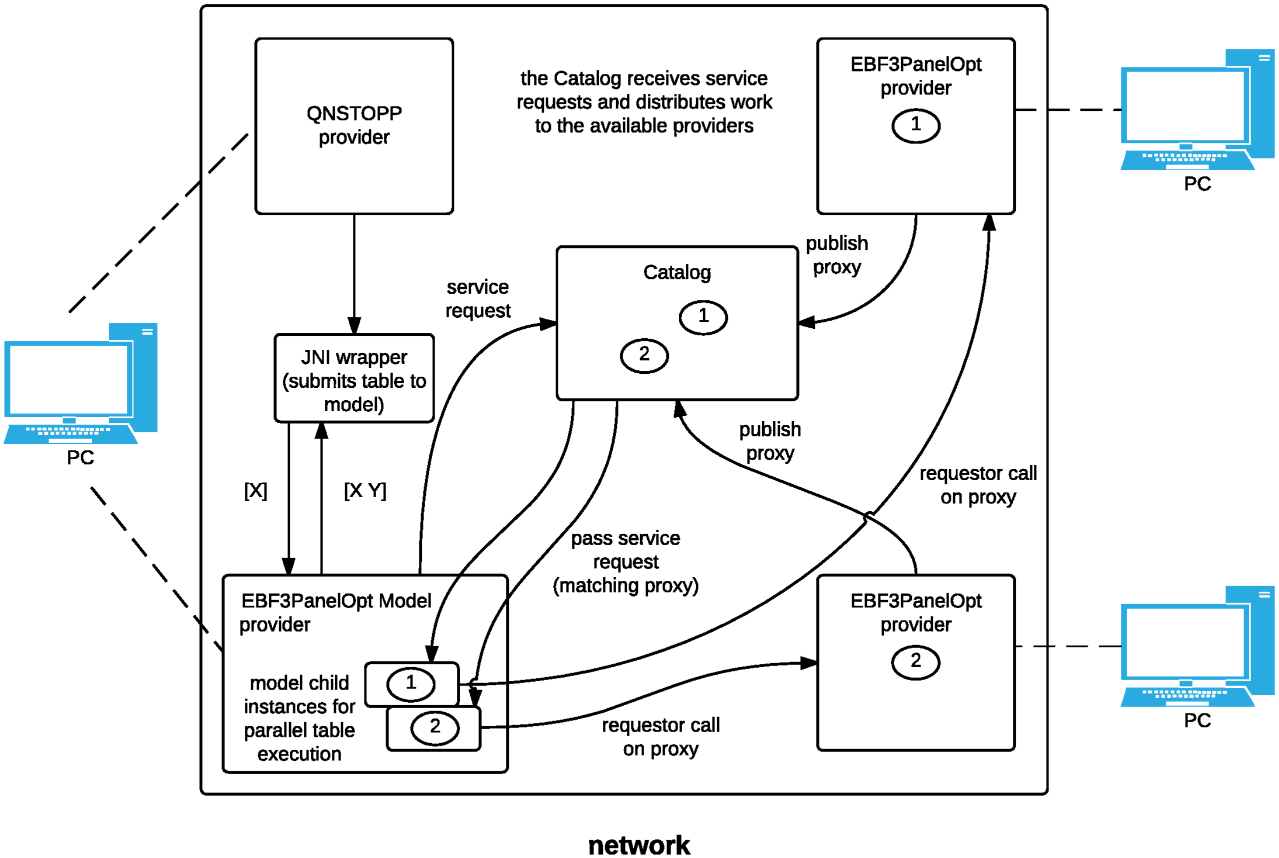
EBF3PanelOpt is an analysis provider distributed over SORCER network using:

- JavaSpaces: exertions dropped into JavaSpace and discovered by providers via Jini



Experiment: Catalog Alternative

SORCER catalog matches services to predefined list of providers



Experiment: Setup

2 identical Intel i7-3770 machines (Ivy Bridge) @3.4 GHz

- Quad-core w/ hyperthreading
- 16 GB memory
- Optimization function & model provider run on one machine, EBF3PanelOpt analysis provider on the other
- For QNSTOPP # threads set to 4
- For VTdirect and QNSTOPS (serial), only 1 analysis provider used at a time

Experiment: Terminology

Parallel efficiency of QNSTOPP w/ and w/o SORCER modelled by:

$$E_p = \frac{\big((\text{QNSTOPS time})/(\text{QNSTOPP time})\big)}{(\text{total number of OMP threads/analysis providers})}.$$

“Script Robustness” is a Java **GenericUtil** for increased robustness of scripts and communication links across different systems

Results: Table 1

Execution times in seconds for pylon wing panel optimization with 2 stiffener panels

	VTdir	pVTdir	QNSTOPS	QNSTOPP	E_p
SORCER and script robustness	13,009	N/A	11,388	3,545	0.80
SORCER w/o script robustness	8,957	N/A	7,994	2,542	0.79
SORCER/Catalog w/o script robust.	8,487	N/A	7,597	2,458	0.77
W/o SORCER, w/o script robust.	8,460	2,924	7,560	2,309	0.82

Results: Table 2

Execution times in seconds for pylon wing panel optimization with 4 stiffener panels

	VTdir	pVTdir	QNSTOPS	QNSTOPP	E_p
SORCER w/ script robustness	14,450	N/A	10,370	3,676	0.71
SORCER w/o script robustness	10,384	N/A	7,451	2,697	0.69
SORCER/Catalog w/o script robust.	9,815	N/A	7,088	2,615	0.68
W/o SORCER, w/o script robust.	9,786	3,789	7,052	2,408	0.73

Results: Table 3

Objective function evaluation times in seconds for pylon wing panel (2 & 4 stiffeners)
Note: 2 stiffeners = 13 dimensional problem, 4 stiffeners = 25 dimensional problem

For 100 function evaluations done through VTdirect, average function evaluation cost:

	$n = 13$	$n = 25$
With SORCER and script robustness	11.13	12.90
With SORCER, without script robustness	7.36	9.14
Without SORCER and script robustness	7.32	9.10

Discussion

Advantages:

- Dynamic distributed resource management
- High level of abstraction, tailored to modelling/design analyses
- Code reusability

Disadvantages:

- Heavyweight (in comparison to Condor, Globus, MPI)
- Overhead of wrapping existing code with JNI

Thanks for Your Time!

Acknowledgements

This material is based on research sponsored by Air Force Research Laboratory under agreement number FA8650-09-2-3938. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes not withstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory or the U.S. Government. The EBF3PanelOpt code was developed under a research contract from NASA Fundamental Aeronautics Program to Virginia Polytechnic Institute and State University with Karen M. B. Taminger as the Program Manager.