Parallel and Distributed Computing with MATLAB

High-performance Signal and Data Processing: Challenges in Astro- and Particle Physics and Radio Astronomy Instrumentation

30 January 2014, Johannesburg

OPTI-NUM solutions

Agenda

- Introduction
- Introduction to Parallel Computing
- Built in Parallel Support
- Composite Arrays and Batch Processing
- Using GPUs
- Built in GPU Support
- GPU Arrays
- Distributed GPU Computing
- Scaling Up to a Cluster
- Concluding Remarks

Welcome

David Gray
Sales Manager

Cheri Meyer-Palmer
Sales Account Manager

Nigel De Bruin
Sales Account Manager

Vicky Makhathini
Sales Account Manager (Education)

Nicole Wilson
Applications Engineer

About OPTI-NUM solutions

Technical Computing and Model-Based Design solutions using MathWorks tools

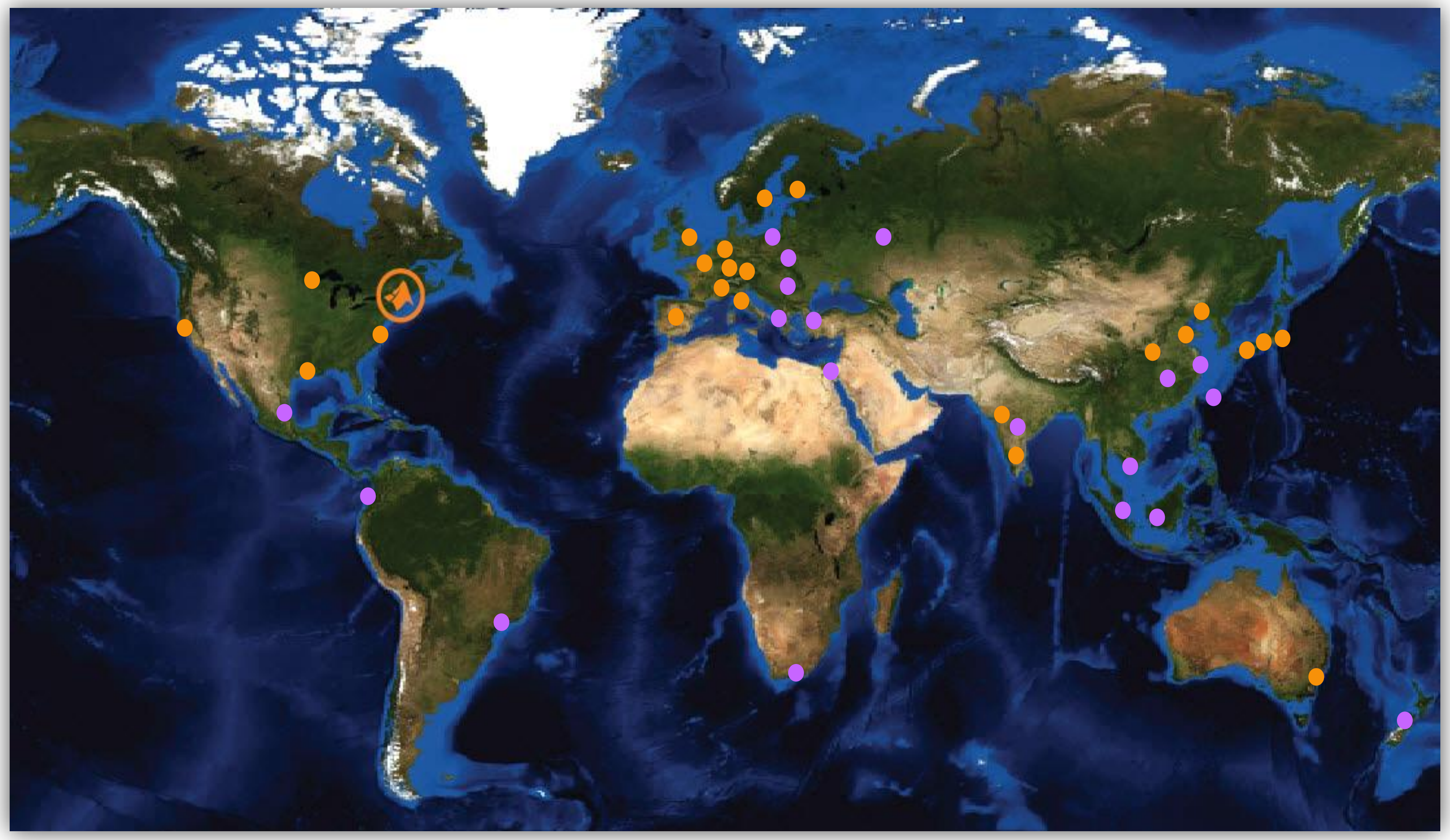
- Sales and support of MathWorks products for Southern Africa
- Consulting and training services

Started by a group of Wits University researchers in 1992

- Founding member is now VP of Development at MathWorks

19 staff, with over 50% technically focussed

MathWorks at a Glance



MathWorks Today

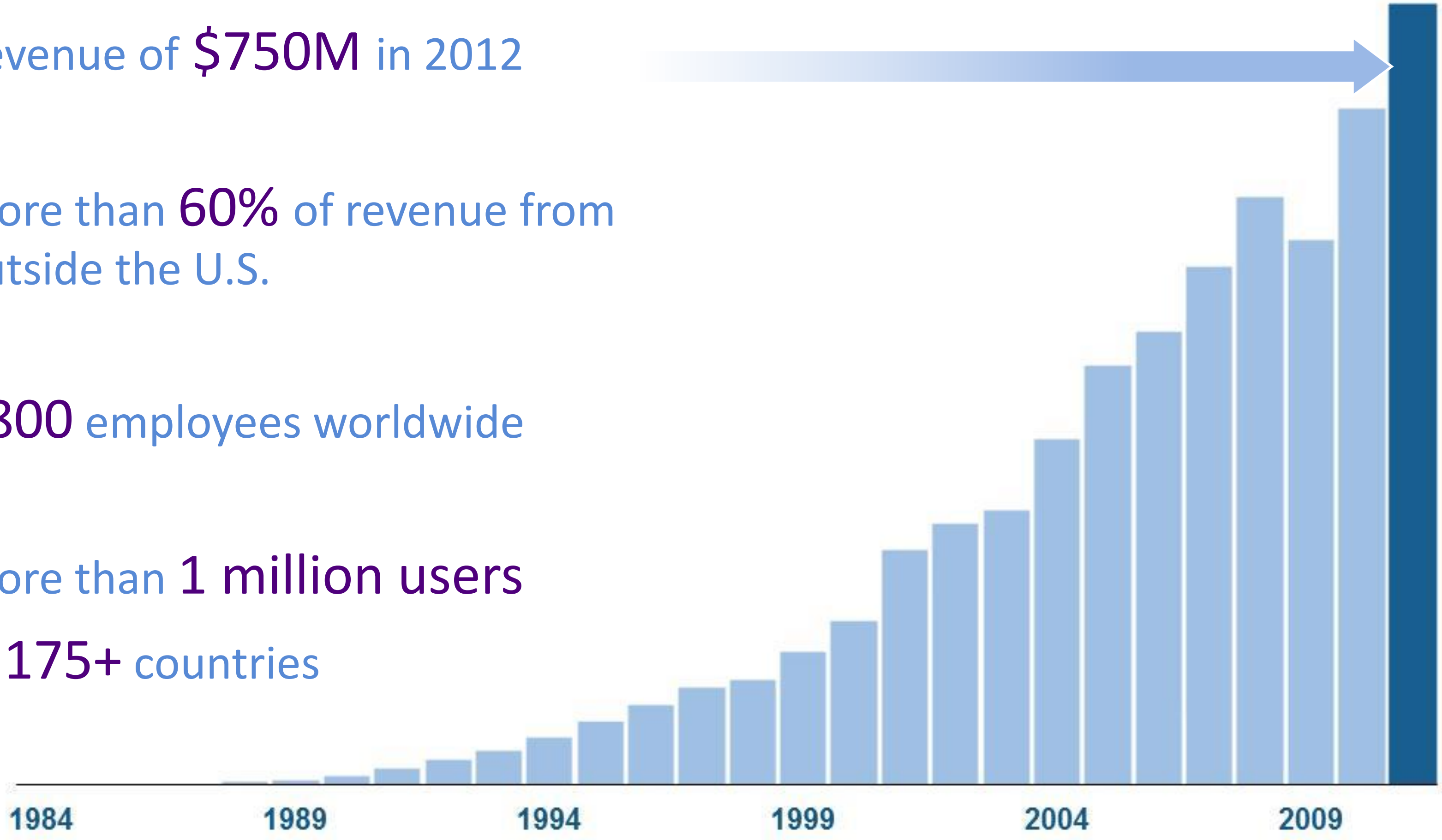
Revenue of $750M in 2012

More than 60% of revenue from outside the U.S.

2800 employees worldwide

More than 1 million users

In 175+ countries



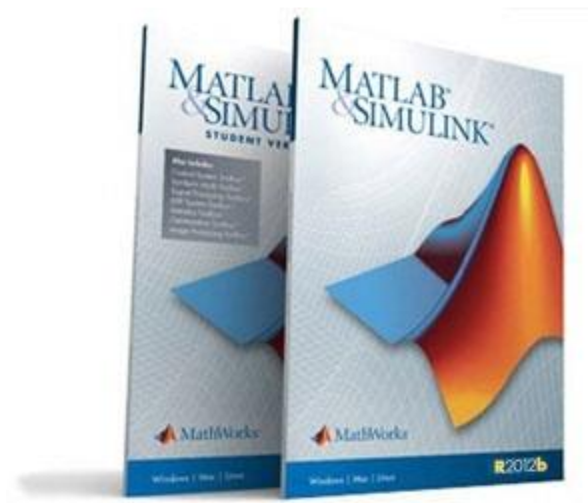
Customer-Driven Business

Two major releases per year

Software improvements driven by customers

All customers subscribed to Software Maintenance Service get:

- Access to every release
- Technical support



R2013b

Key Industries

Aerospace and Defense

Automotive

Biotech and Pharmaceutical

Communications

Education

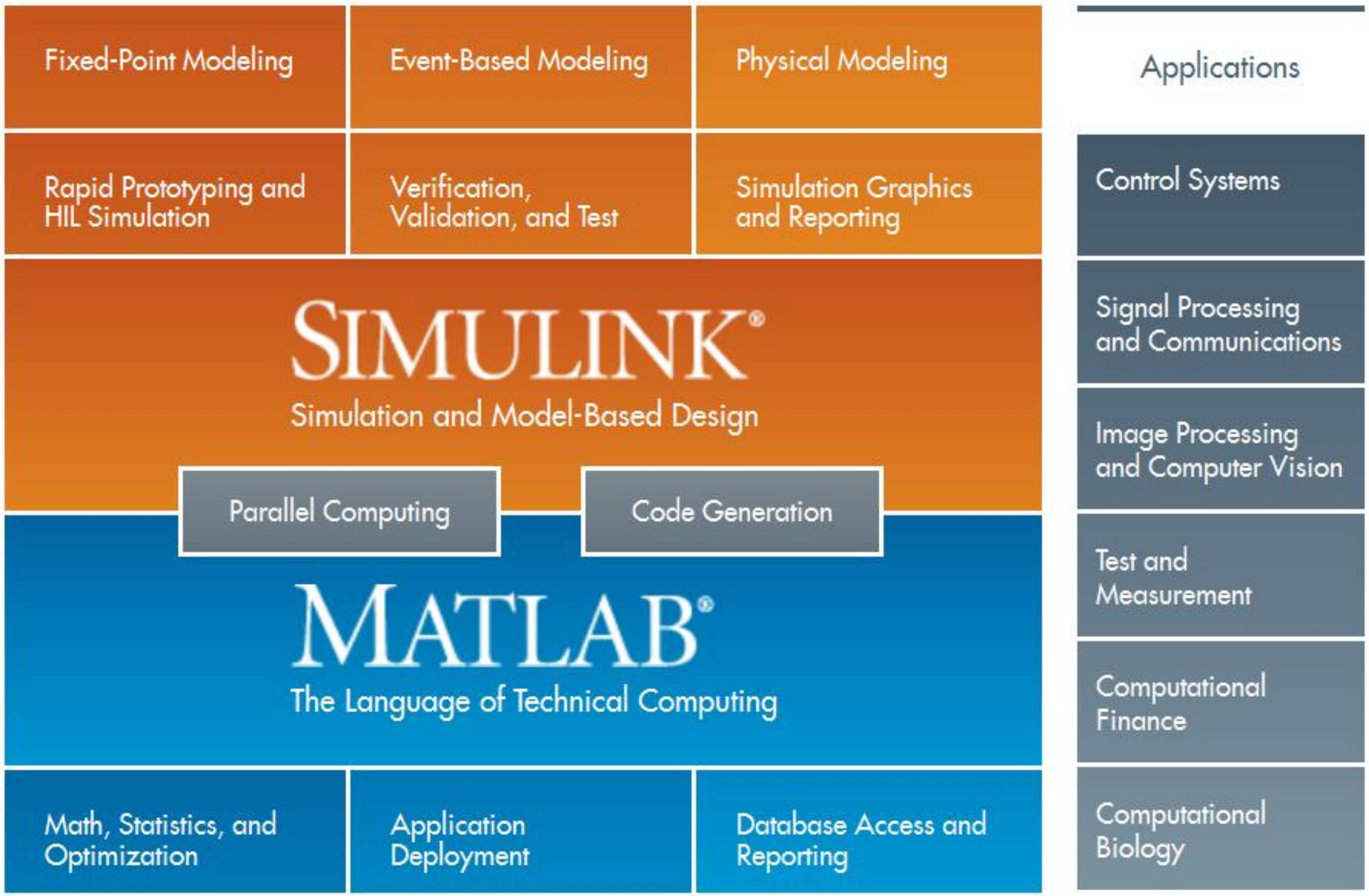
Electronics and Semiconductors

Energy Production

Financial Services

Industrial Automation and Machinery

MathWorks Product Overview



Agenda

- Introduction
- Introduction to Parallel Computing
- Built in Parallel Support
- Composite Arrays and Batch Processing
- Using GPUs
- Built in GPU Support
- GPU Arrays
- Distributed GPU Computing
- Scaling Up to a Cluster
- Concluding Remarks

Solving big technical problems

Challenges	You could…		Solutions
Long running Computationally intensive	Wait	⇨	Larger compute pool (E.G. More processors)
Large data set	Reduce size of problem	⇨	Larger memory pool (E.G. More machines)

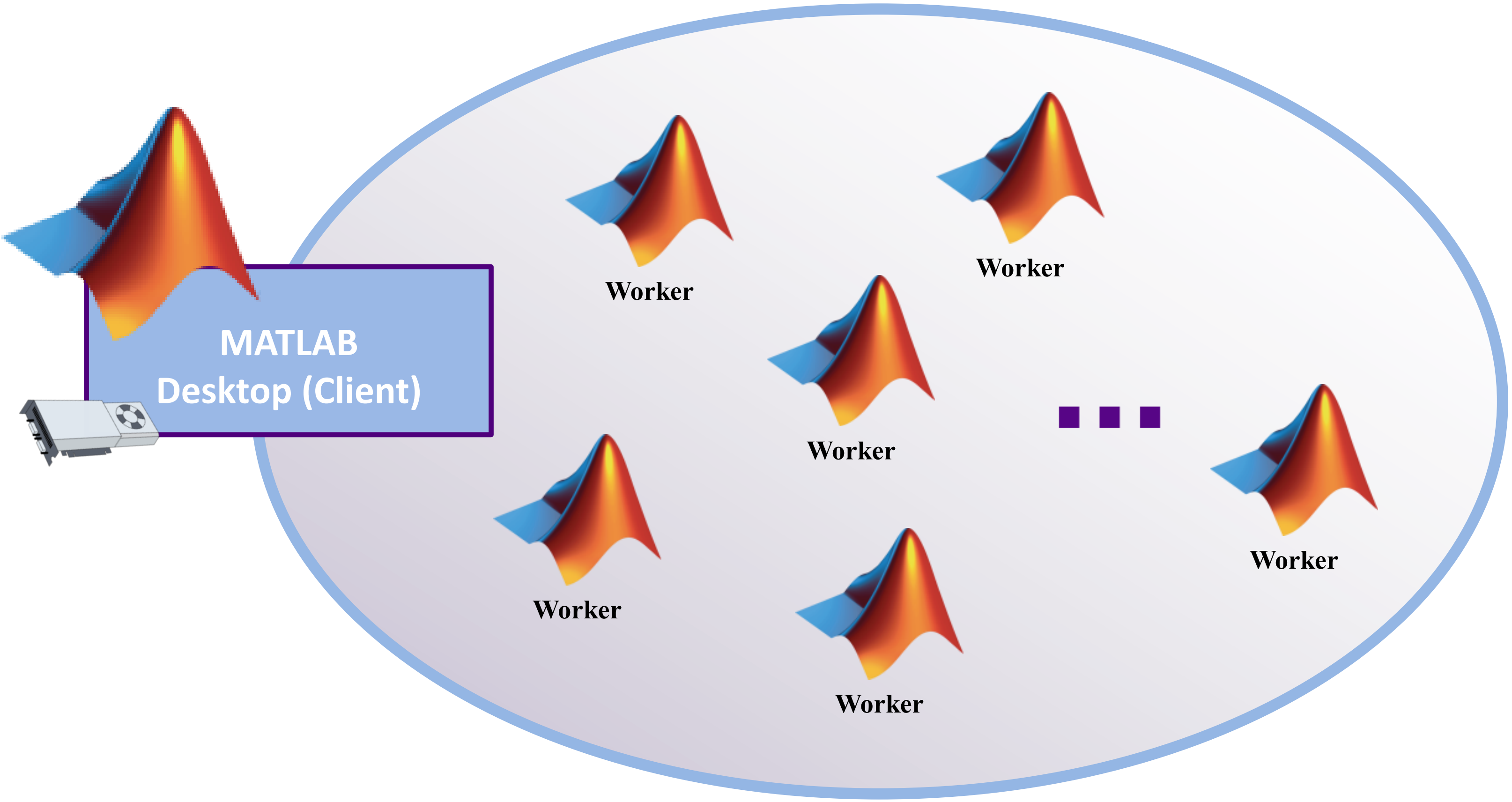
Utilizing Additional Processing Power

- Built-in multithreading (implicit)
  - Core MATLAB
  - Introduced in R2007a
  - Utility for specific matrix operations
  - Multiple threads in a single MATLAB computation engine
  - Automatically enabled since R2008a

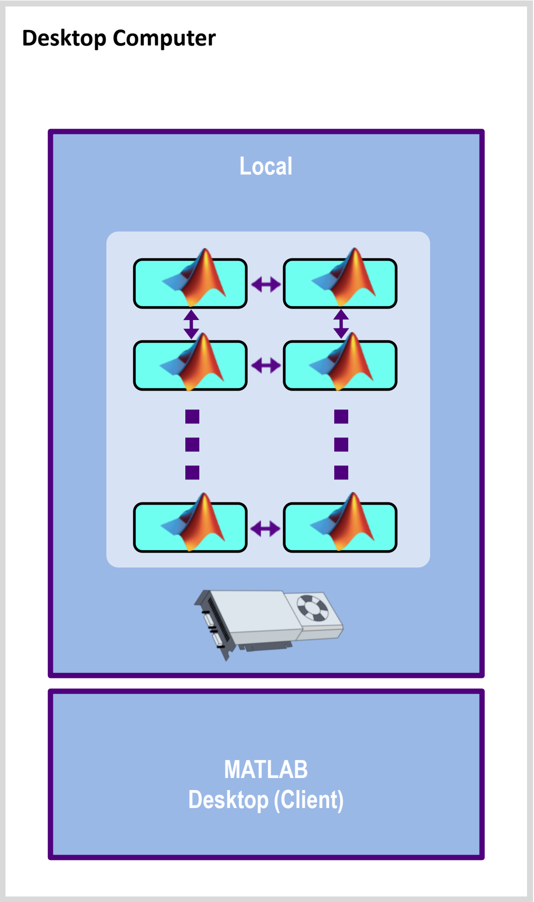
www.mathworks.com/discovery/multicore-matlab.html

- Parallel computing tools (explicit)
  - Parallel Computing Toolbox
  - MATLAB Distributed Computing Server
  - Broad utility controlled by the MATLAB user

Going Beyond Serial MATLAB Applications

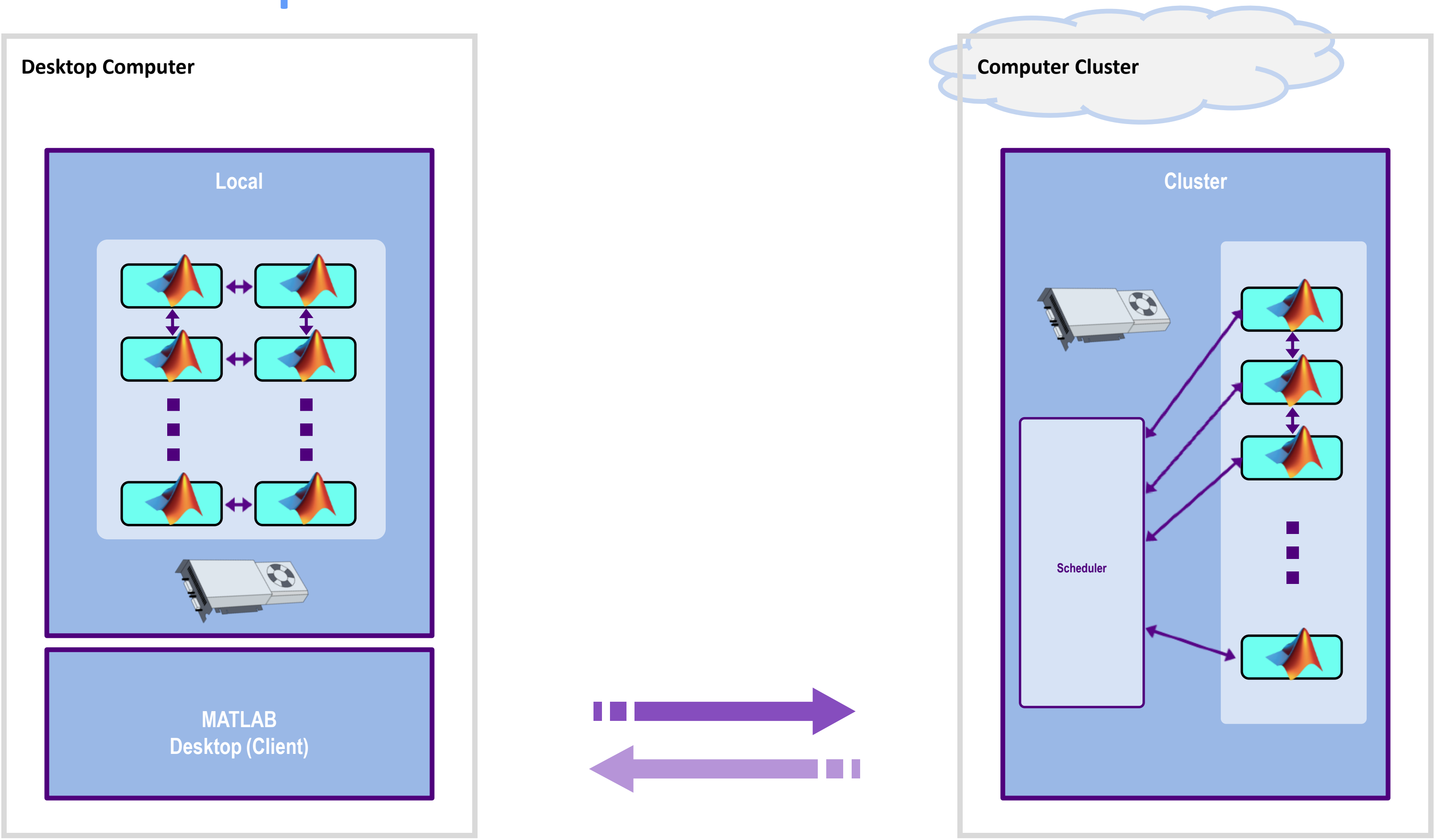


Parallel Computing Toolbox for the Desktop



- Speed up parallel applications
- Take advantage of GPUs
- Prototype code for your cluster

Scale Up to Clusters and Clouds



Considerations

What is the data transfer time

- Another machine in same room?
- Another machine in another country?
- Connection type?

What is your required speedup?

Are you sharing the cluster with other people?

What would be involved in parallelising your algorithm?

Programming Parallel Applications

Level of control	*Required effort*
Minimal	***None***
Some	***Straightforward***
Extensive	***Involved***

Programming Parallel Applications

Level of Control	CPU Parallel	GPU
Minimal	Support built into toolboxes	Built-in GPU support (support built into existing functions, GPU arrays)
Some	High-level programming constructs (`parfor`, batch, distributed arrays, composites)	Execute custom functions on the GPU array
Extensive	Low-level programming constructs (jobs/tasks, `spmd`)	Invoke CUDA kernels directly from MATLAB
	Distributed GPU Computing	

Agenda

- Introduction
- Introduction to Parallel Computing
- Built in Parallel Support
- Composite Arrays and Batch Processing
- Using GPUs
- Built in GPU Support
- GPU Arrays
- Distributed GPU Computing
- Scaling Up to a Cluster
- Concluding Remarks

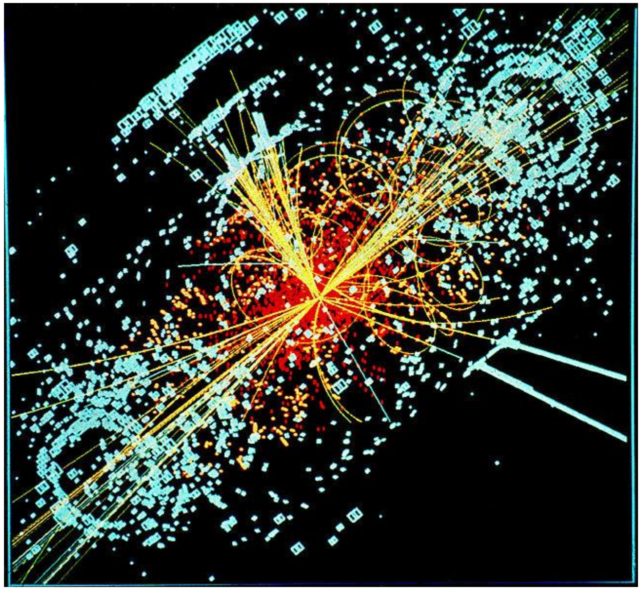
Programming Parallel Applications

Level of Control	CPU Parallel	GPU
Minimal	**Support built into toolboxes**	Built-in GPU support (support built into existing functions, GPU arrays)
Some	High-level programming constructs (`parfor`, batch, distributed arrays, composites)	Execute custom functions on the GPU array
Extensive	Low-level programming constructs (jobs/tasks, `spmd`)	Invoke CUDA kernels directly from MATLAB
	Distributed GPU Computing	

Example – Neural Networks

Aim: Use a Neural Network to classify two types of particles generated in high energy collider experiments, based different attributes:

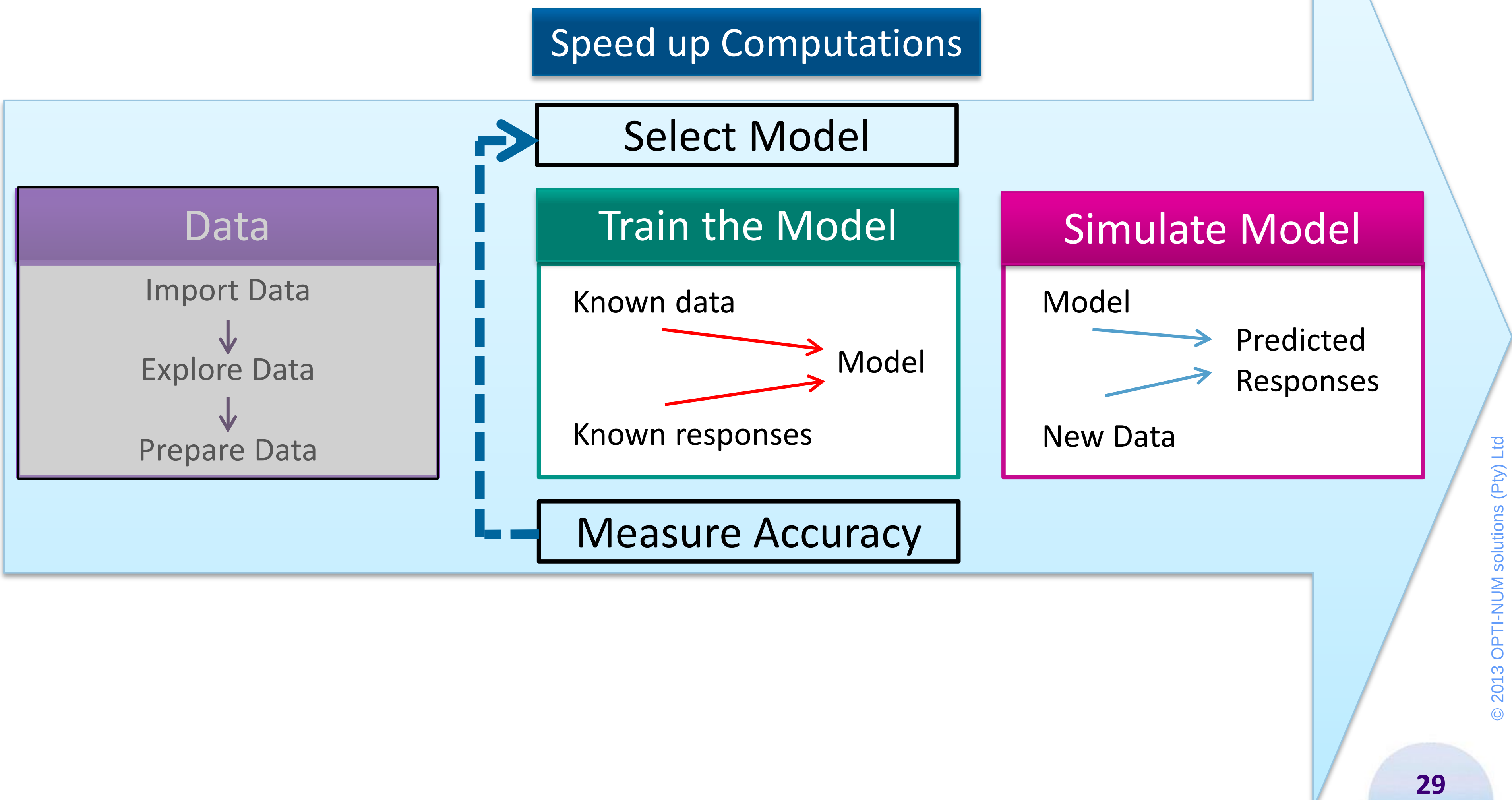
- Binary classification problem
- 78 different attributes (momentum, direction, mass, curvature, etc.)
- 50 000 examples in the dataset



Simulated particle collision in the Large Hadron Collider (LHC)

Acknowledgement of dataset: http://osmot.cs.cornell.edu/kddcup/faqs.html

Supervised Learning Workflow



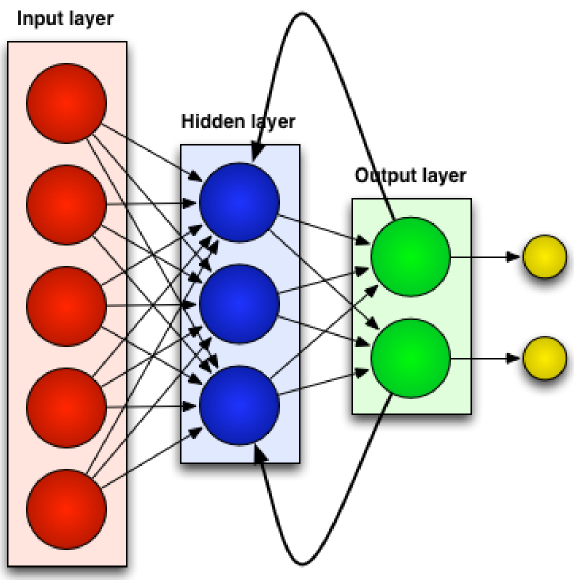
Example – Neural Networks

Utilising Built-In Support

```
% Train and simulate the neural network with data split by sample across
% all the workers
net2 = train(net1,x,t,'useParallel','yes');
y = net2(x,'useParallel','yes');
```

Goals:

- Distribute the training dataset across workers
- Distribute the simulation dataset across workers
- Parallelise training and simulation with the least possible effort



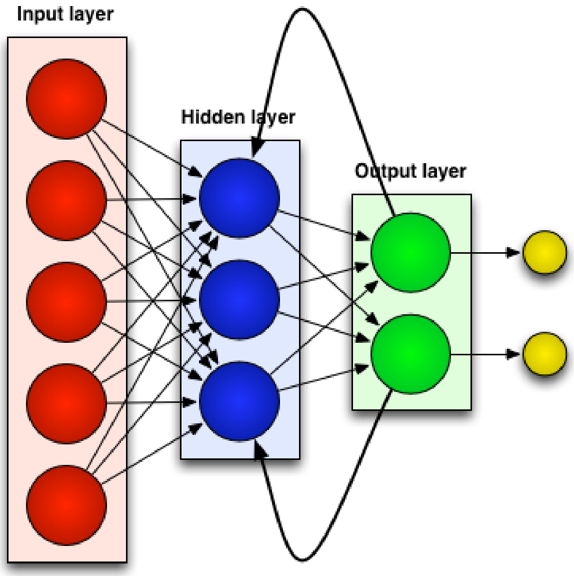
Example – Neural Networks

Utilising Built-In Support

```
% Train and simulate the neural network with data split by sample across
% all the workers
net2 = train(net1,x,t,'useParallel','yes');
y = net2(x,'useParallel','yes');
```

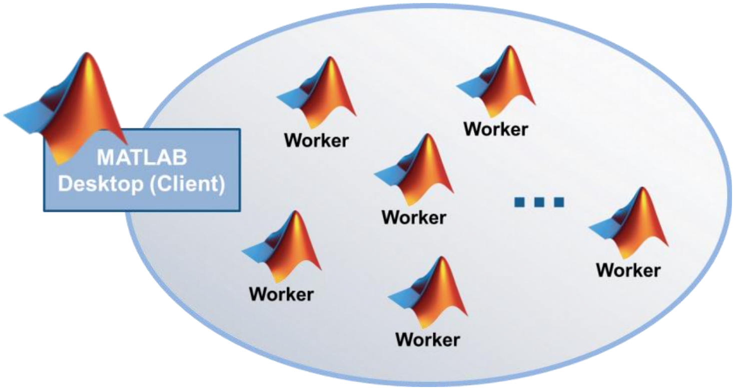
Summary:

- Set "useParallel" to "yes" to make use of built-in parallelisation
- Dataset distributed evenly across workers
- No knowledge of parallel computing required



Tools Providing Parallel Computing Support

- Optimization Toolbox, Global Optimization Toolbox
- Statistics Toolbox
- Signal Processing Toolbox
- Neural Network Toolbox
- Image Processing Toolbox
- …



Directly leverage functions in Parallel Computing Toolbox

www.mathworks.com/products/parallel-computing/builtin-parallel-support.html

Agenda

- Introduction
- Introduction to Parallel Computing
- Built in Parallel Support
- Composite Arrays and Batch Processing
- Using GPUs
- Built in GPU Support
- GPU Arrays
- Distributed GPU Computing
- Scaling Up to a Cluster
- Concluding Remarks

Programming Parallel Applications

Level of Control	CPU Parallel	GPU
Minimal	**Support built into toolboxes**	Built-in GPU support (support built into existing functions, GPU arrays)
Some	High-level programming constructs (`parfor`, **batch,** distributed arrays, **composites**)	Execute custom functions on the GPU array
Extensive	Low-level programming constructs (jobs/tasks, `spmd`)	Invoke CUDA kernels directly from MATLAB
	Distributed GPU Computing	

Composites

- Data type for which variables are created in the client session but data is created directly on the workers
- Represent cell arrays in their display and usage
- Use to prepopulate values on workers before training starts

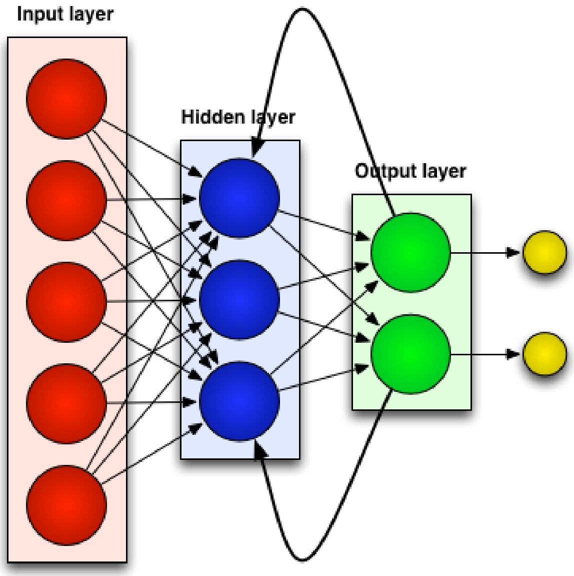
Example – Neural Networks

Manually divide data across the workers

```
xc = Composite;
tc = Composite;
```

Motivation:

- Problem size is too big for the host computer
- Some workers are on computers that are faster or have more memory than others



Goal:

- Use Composites to specify which subsets of the data should go on which workers

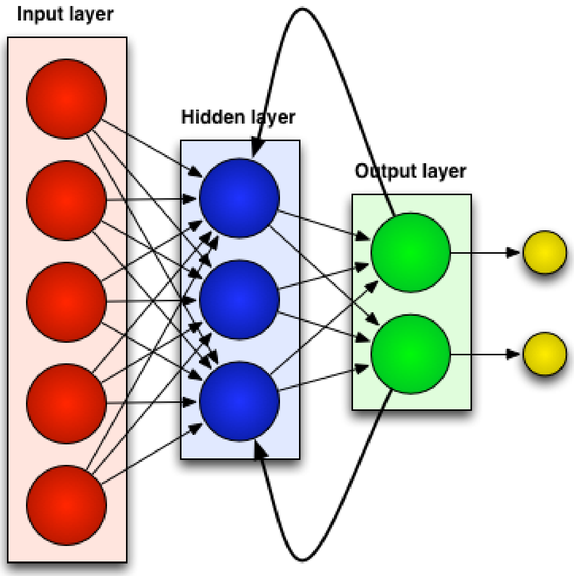
Example – Neural Networks

Manually divide data across the workers

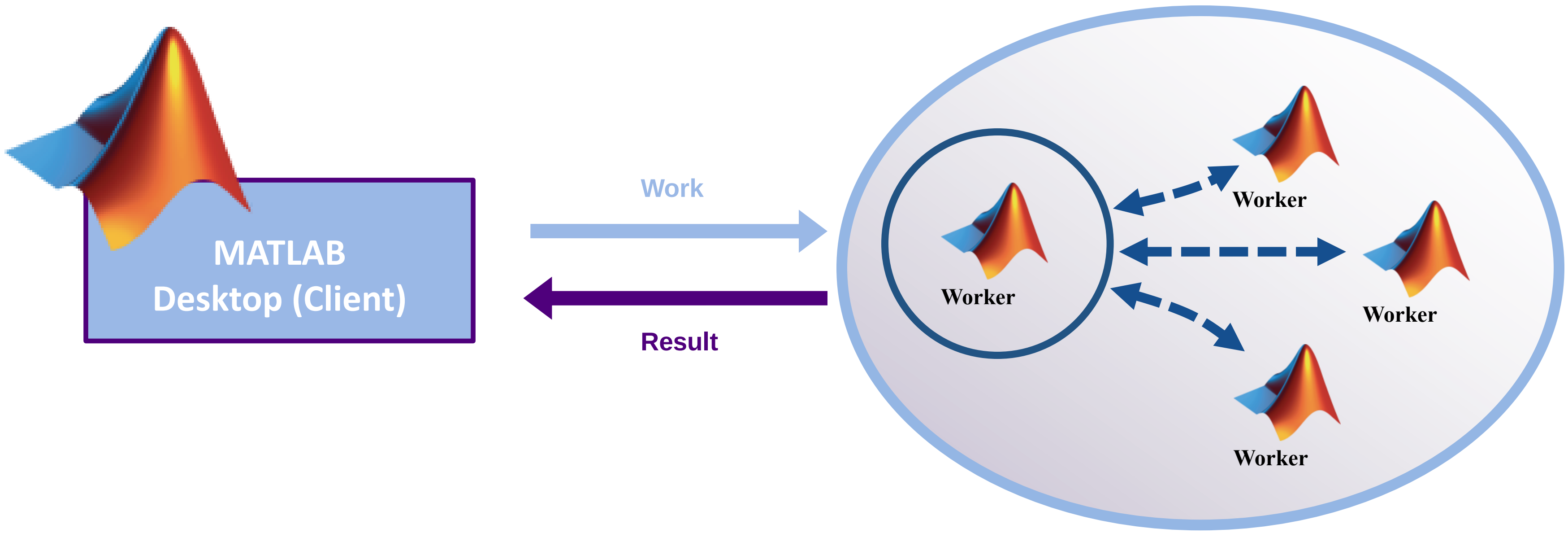
```
xc = Composite;
tc = Composite;
```

Summary:

- Composites can be used to specify which subsets of the dataset should reside on which workers
- Manually defining the elements of Composite values sequentially allows much bigger problems to be defined
- You can perform load balancing by distributing the data with differing numbers of samples per worker



Offload Computations with `batch`

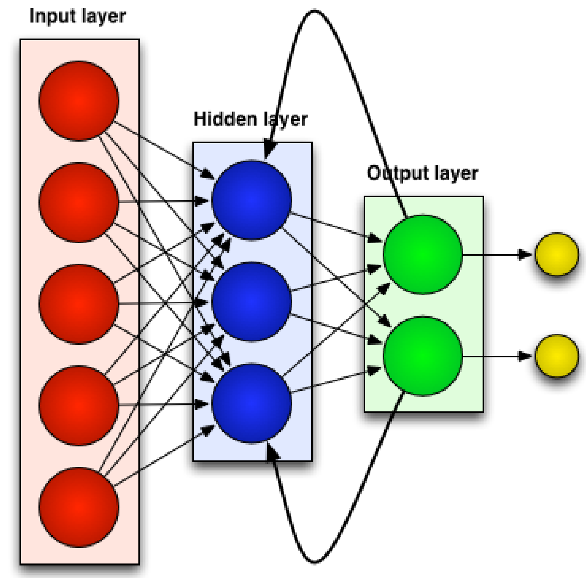


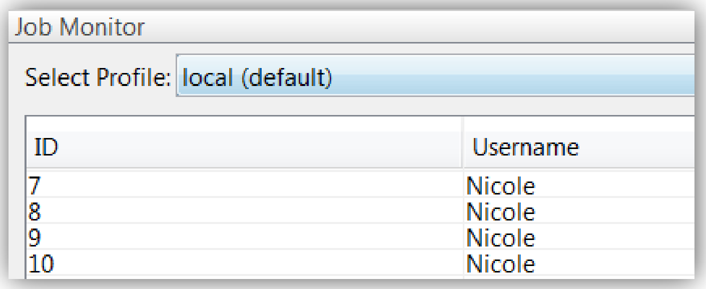
Example – Neural Networks

Offload data to workers using BATCH

Goal:

- Interactively and programmatically offload computations to one or more workers
- Load results back from workers when computations are complete





Job Monitor

Select Profile: local (default)

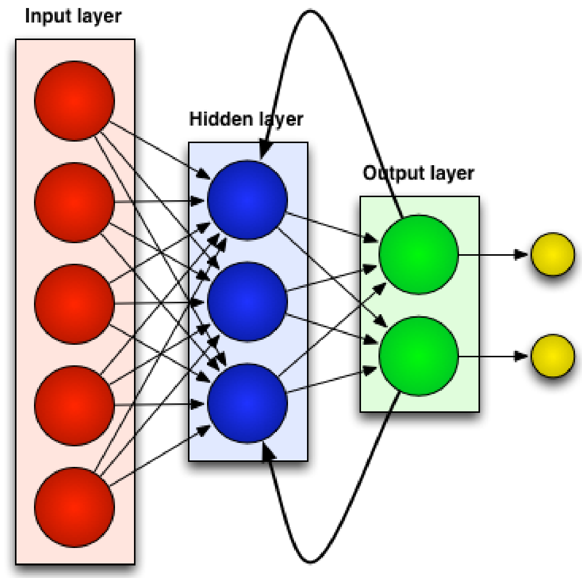
ID	Username
7	Nicole
8	Nicole
9	Nicole
10	Nicole

Example – Neural Networks

Offload data to workers using BATCH

Summary:

- Interactively offload computations to another worker to free up the client session of MATLAB for other work
- Run offloaded computation in parallel by making use of additional workers
- Job manager allows you to see what jobs are running

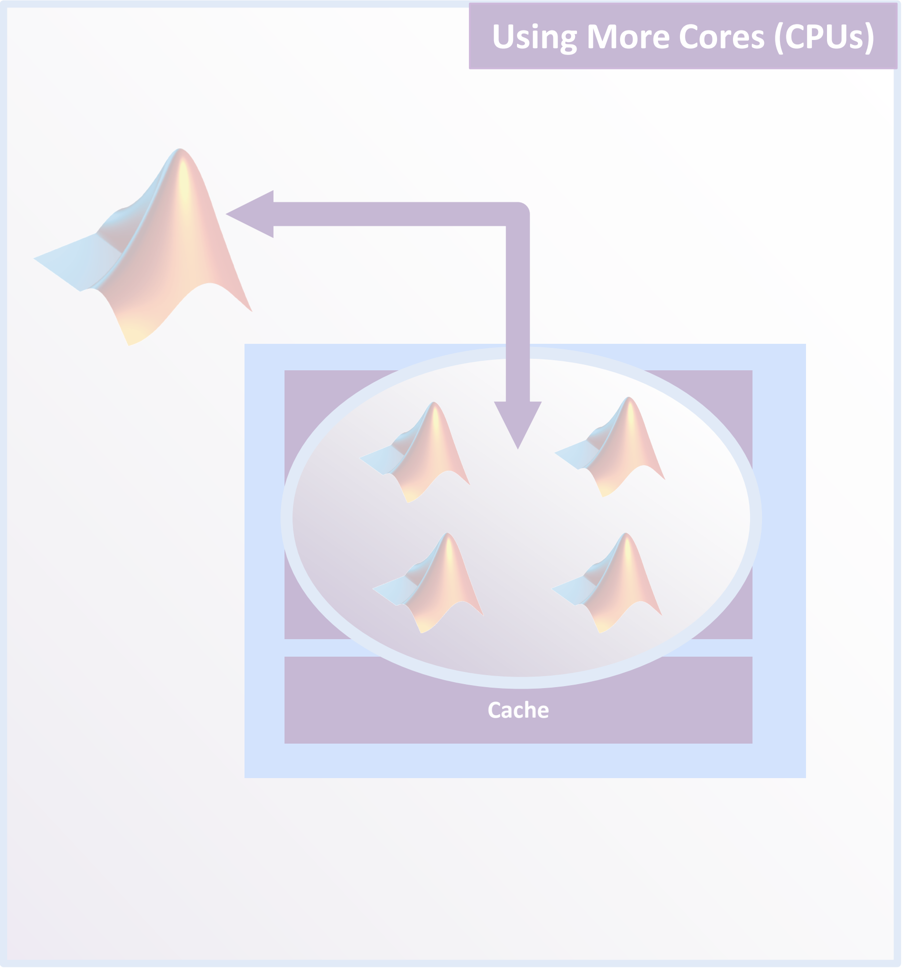


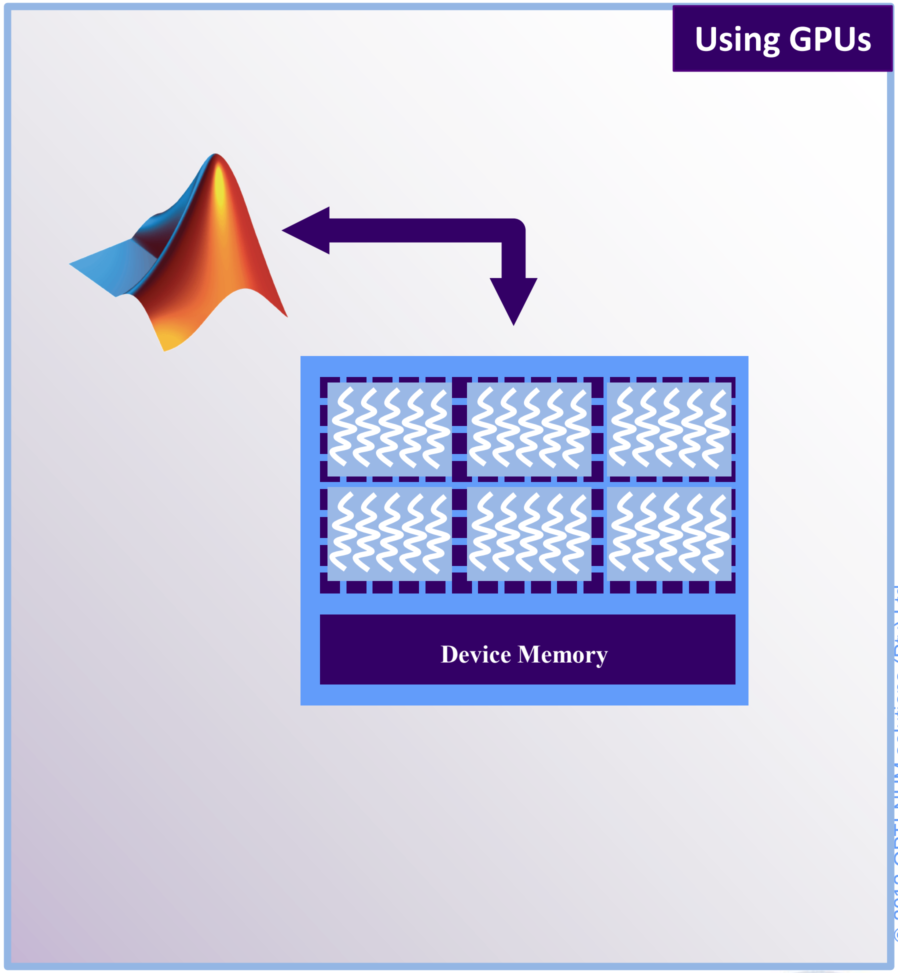
Job Monitor

Select Profile: local (default)

ID	Username
7	Nicole
8	Nicole
9	Nicole
10	Nicole

Performance Gain with More Hardware





Agenda

- Introduction
- Introduction to Parallel Computing
- Built in Parallel Support
- Composite Arrays and Batch Processing
- Using GPUs
- Built in GPU Support
- GPU Arrays
- Distributed GPU Computing
- Scaling Up to a Cluster
- Concluding Remarks

What is a Graphics Processing Unit (GPU)

- Originally for graphics acceleration, now also used for scientific calculations
- Massively parallel array of integer and floating point processors
  - Typically hundreds of processors per card
  - GPU cores complement CPU cores
- Dedicated high-speed memory

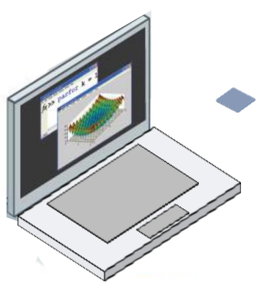
New: GPU functionality in R2010b

* Parallel Computing Toolbox requires NVIDIA GPUs with Compute Capability 1.3 or higher, including NVIDIA Tesla 20-series products. See a complete listing at www.nvidia.com/object/cuda_gpus.html

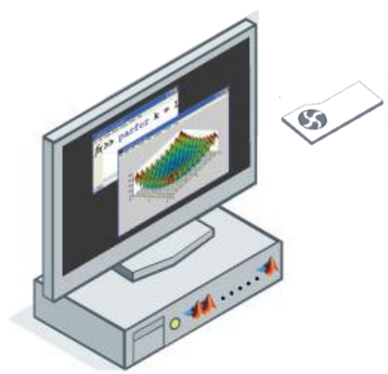
GPU Performance – not all cards are equal

- Tesla-based cards will provide best performance
- Realistically, expect 4x to 15x speedup (Tesla) vs CPU
- See GPUBench on MATLAB Central for examples

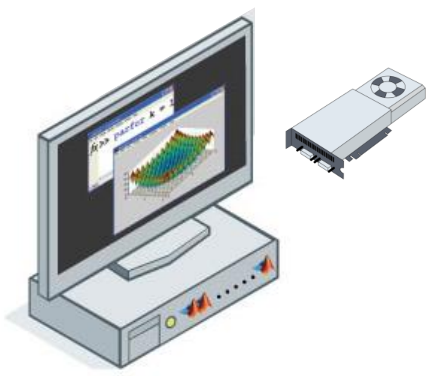
www.mathworks.com/matlabcentral/fileexchange/34080-gpubench



Laptop
GPU
GeForce



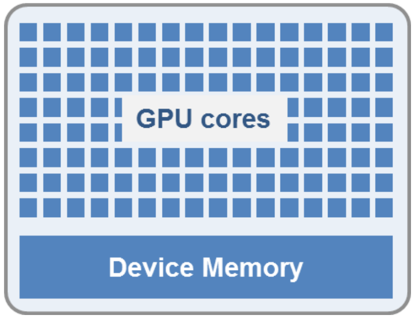
Desktop GPU
GeForce /
Quadro



High Performance
Computing GPU
Tesla / Quadro

Criteria for Good Problems to Run on a GPU

- **Massively parallel:**
  - Calculations can be broken into hundreds or thousands of independent units of work
  - Problem size takes advantage of many GPU cores



- **Computationally intensive:**
  - Computation time significantly exceeds CPU/GPU data transfer time

- **Algorithm consists of supported functions:**
  - Growing list of Toolboxes with built-in support
    - www.mathworks.com/products/parallel-computing/builtin-parallel-support.html
  - Subset of core MATLAB for `gpuArray, arrayfun, bsxfun`
    - www.mathworks.com/help/distcomp/using-gpuarray.html#bsloua3-1
    - www.mathworks.com/help/distcomp/execute-matlab-code-elementwise-on-a-gpu.html#bsnx7h8-1

Programming Parallel Applications

Level of Control	CPU Parallel	GPU
Minimal	Support built into toolboxes	Built-in GPU support (support built into existing functions, GPU arrays)
Some	High-level programming constructs (`parfor`, batch, distributed arrays, composites)	Execute custom functions on the GPU array
Extensive	Low-level programming constructs (jobs/tasks, `spmd`)	Invoke CUDA kernels directly from MATLAB
	Distributed GPU Computing	

Programming Parallel Applications

Level of Control	CPU Parallel	GPU
Minimal	Support built into toolboxes	**Built-in GPU support (support built into existing functions, GPU arrays)**
Some	High-level programming constructs (`parfor`, batch, distributed arrays, composites)	Execute custom functions on the GPU array
Extensive	Low-level programming constructs (jobs/tasks, `spmd`)	Invoke CUDA kernels directly from MATLAB
	Distributed GPU Computing	

Agenda

- Introduction
- Introduction to Parallel Computing
- Built in Parallel Support
- Composite Arrays and Batch Processing
- Using GPUs
- Built in GPU Support
- GPU Arrays
- Distributed GPU Computing
- Scaling Up to a Cluster
- Concluding Remarks

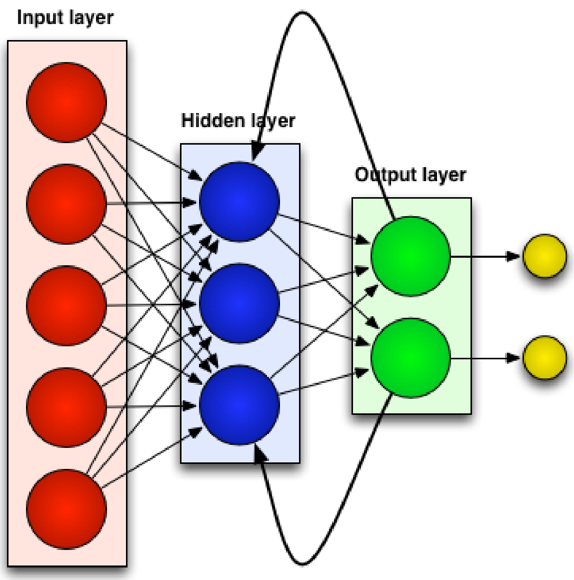
Example – Neural Networks

GPU Computing using support built Into existing functions

```
net2 = train(net1,x,t,'useGPU','yes');
```

Goals:

- Utilise dedicate high-speed memory and highly parallel nature of GPU to train and simulate neural network
- Perform GPU programming with little to no knowledge of GPUs



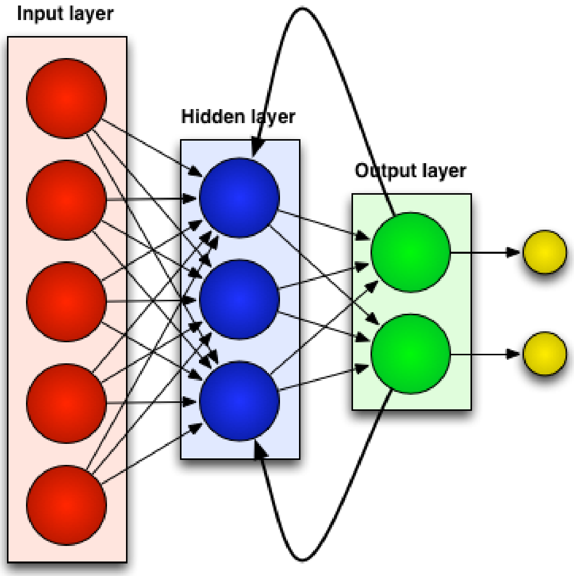
Example – Neural Networks

GPU Computing using support built
Into existing functions

```
net2 = train(net1,x,t,'useGPU','yes');
```

Summary:

- Set “useGPU” flag to “yes” to utilise GPU support built into existing functions
- No knowledge of GPUs required



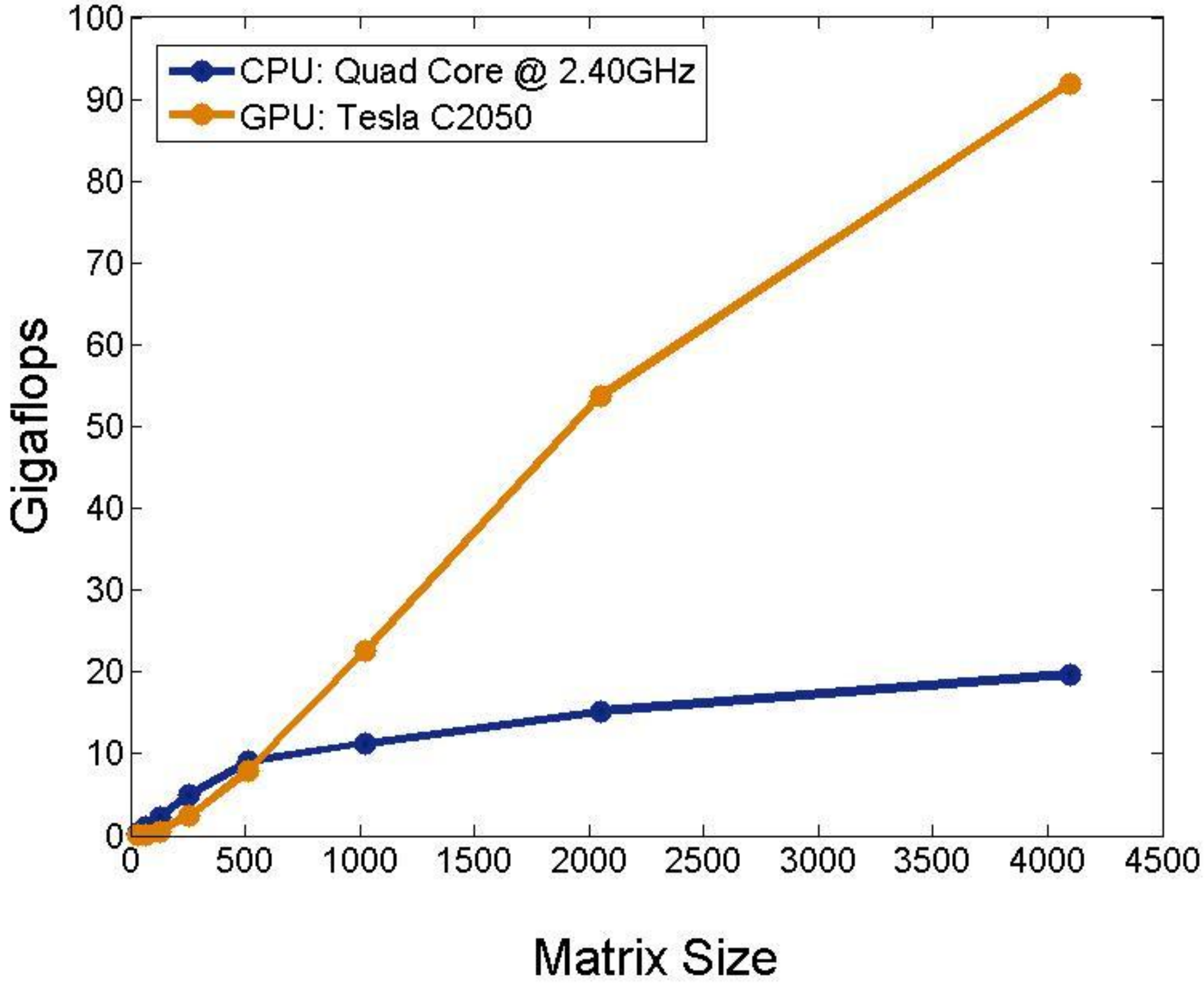
Agenda

- Introduction
- Introduction to Parallel Computing
- Built in Parallel Support
- Composite Arrays and Batch Processing
- Using GPUs
- Built in GPU Support
- GPU Arrays
- Distributed GPU Computing
- Scaling Up to a Cluster
- Concluding Remarks

GPU Arrays

```
>> A = someArray(1000, 1000);
>> G = gpuArray(A); % Push to GPU memory
…
>> F = fft(G);
>> x = G\b;
…
>> z = gather(x); % Bring back into
   MATLAB
```

Performance: A\b with Double Precision



Programming Parallel Applications

Level of Control	CPU Parallel	GPU
Minimal	Support built into toolboxes	Built-in GPU support (support built into existing functions, GPU arrays)
Some	High-level programming constructs (`parfor`, batch, distributed arrays, composites)	Execute custom functions on the GPU array
Extensive	Low-level programming constructs (jobs/tasks, `spmd`)	Invoke CUDA kernels directly from MATLAB
	Distributed GPU Computing	

Programming Parallel Applications (GPU)

Ease of Use

Advanced programming constructs:
`arrayfun, bsxfun, spmd`

Interface for experts:
`CUDAKernel, MEX support`

Greater Control

www.mathworks.com/help/releases/R2013a/distcomp/executing-cuda-or-ptx-code-on-the-gpu.html

www.mathworks.com/help/releases/R2013a/distcomp/create-and-run-mex-files-containing-cuda-code.html

Agenda

- Introduction
- Introduction to Parallel Computing
- Built in Parallel Support
- Composite Arrays and Batch Processing
- Using GPUs
- Built in GPU Support
- GPU Arrays
- Distributed GPU Computing
- Scaling Up to a Cluster
- Concluding Remarks

Programming Parallel Applications

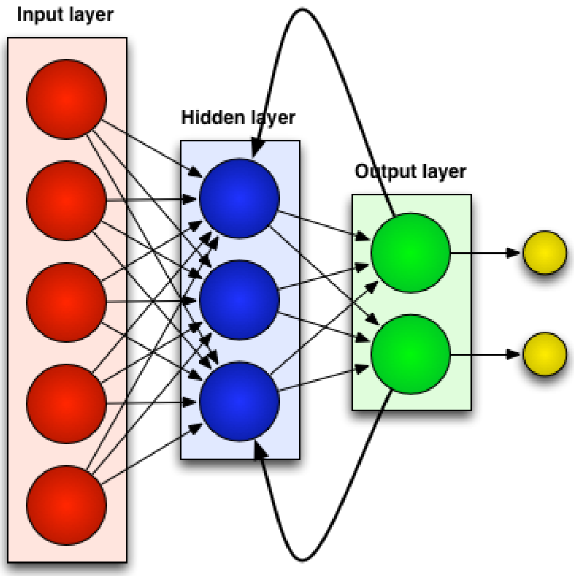
Level of Control	CPU Parallel	GPU
Minimal	Support built into toolboxes	Built-in GPU support (support built into existing functions, GPU arrays)
Some	High-level programming constructs (`parfor`, batch, distributed arrays, composites)	Execute custom functions on the GPU array
Extensive	Low-level programming constructs (jobs/tasks, `spmd`)	Invoke CUDA kernels directly from MATLAB
	Distributed GPU Computing	

Example – Neural Networks

Distributed GPU Computing

Goals:

- Combine multicore computing and GPU computing to run calculations across multiple GPUs and/or CPUs



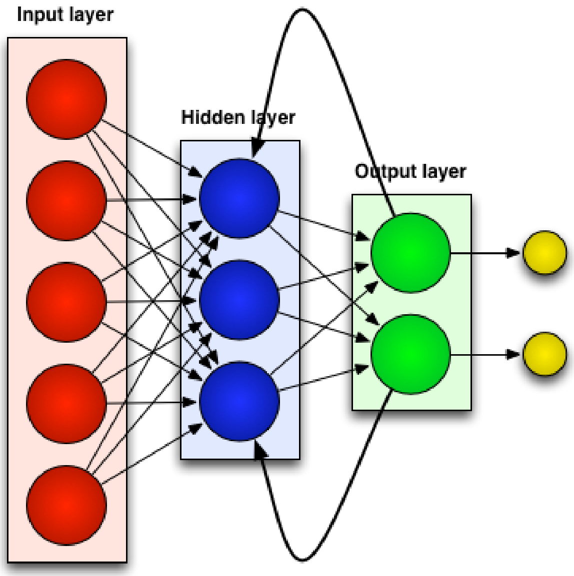

```
net2 = train(net1,x,t,'useParallel','yes','useGPU','yes','showResources','yes')
```

Example – Neural Networks

Distributed GPU Computing

Summary:

- Multicore computing and GPU computing can be combined to run calculations across multiple GPUs and/or CPUs
- GPUs and CPUs can be on a single PC or on clusters of PCs
- Use built-in support or divide data manually using Composites and SPMD

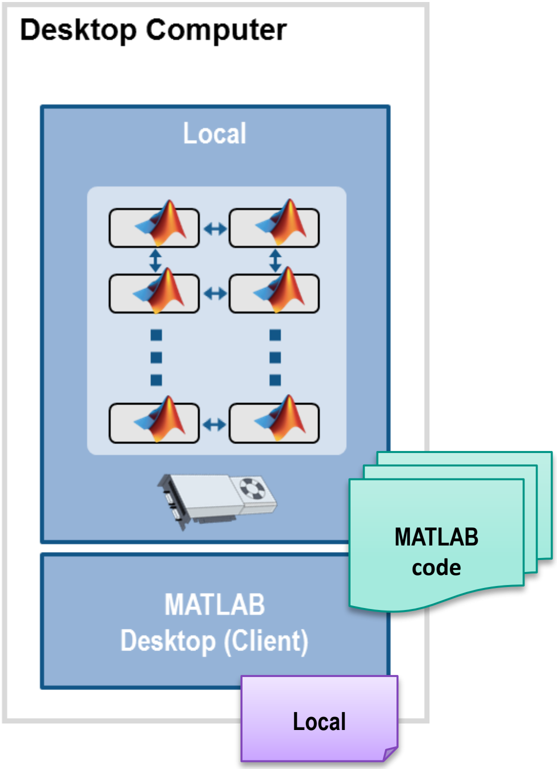



```
net2 = train(net1,x,t,'useParallel','yes','useGPU','yes','showResources','yes')
```

Agenda

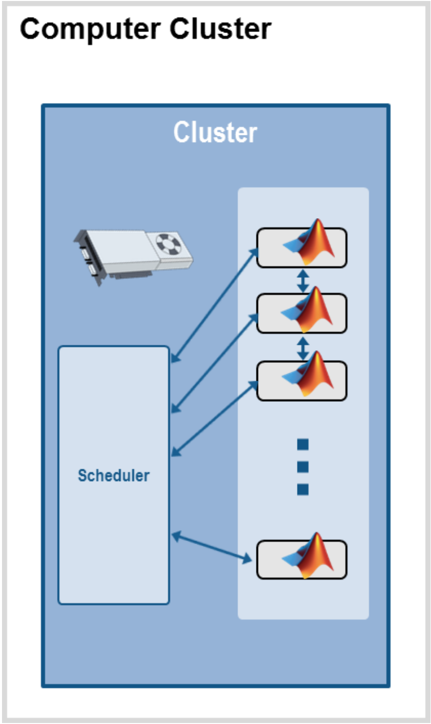
- Introduction
- Introduction to Parallel Computing
- Built in Parallel Support
- Composite Arrays and Batch Processing
- Using GPUs
- Built in GPU Support
- GPU Arrays
- Distributed GPU Computing
- Scaling Up to a Cluster
- Concluding Remarks

Use MATLAB Distributed Computing Server



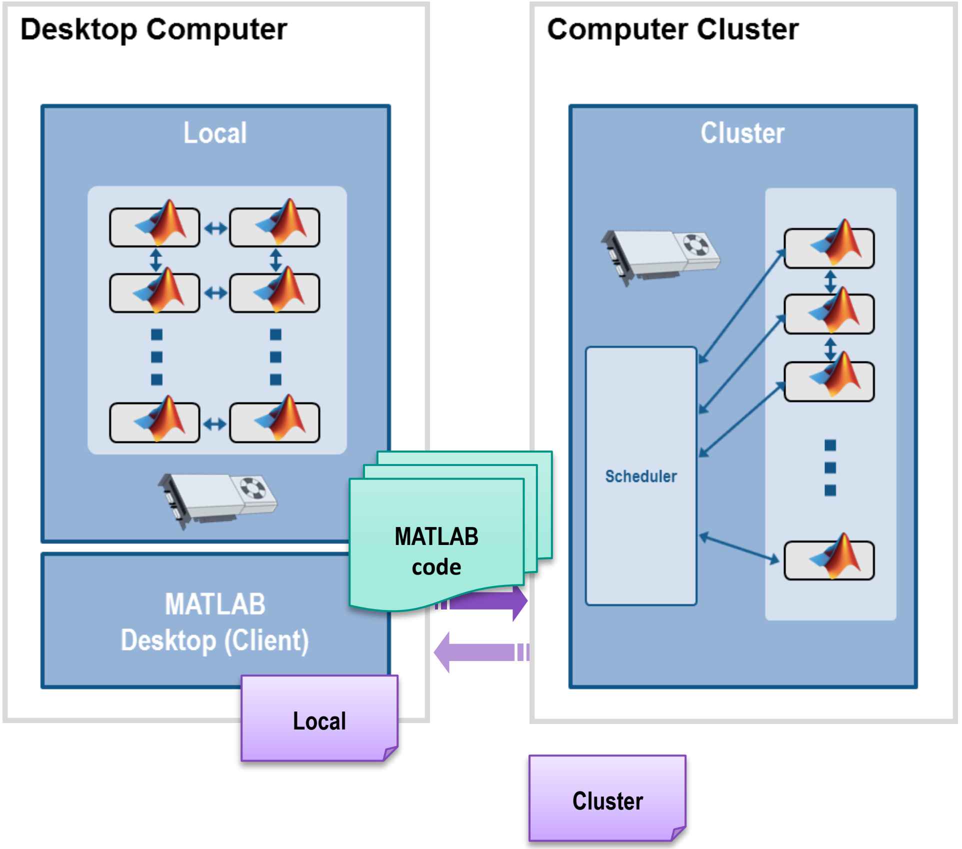
1. Prototype code

Use MATLAB Distributed Computing Server



1. Prototype code
2. Get access to an enabled cluster

Use MATLAB Distributed Computing Server

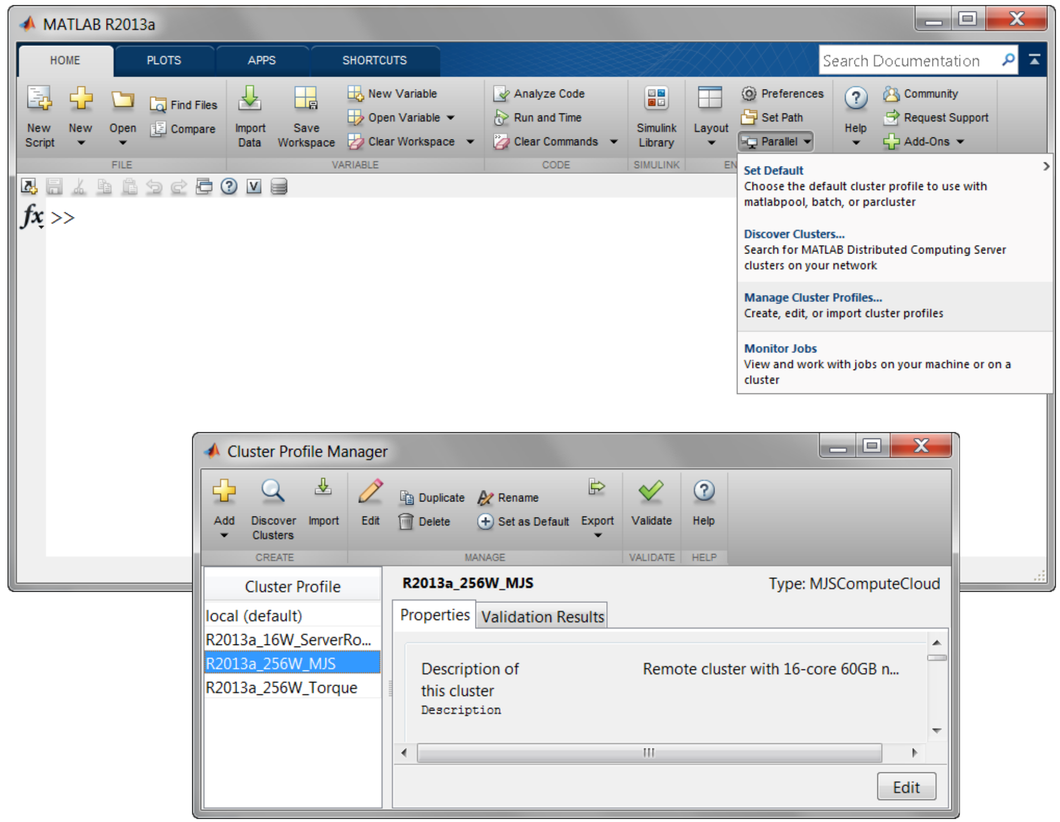


1. Prototype code
2. Get access to an enabled cluster
3. Switch cluster profile to run on cluster resources

Migrate from Desktop to Cluster

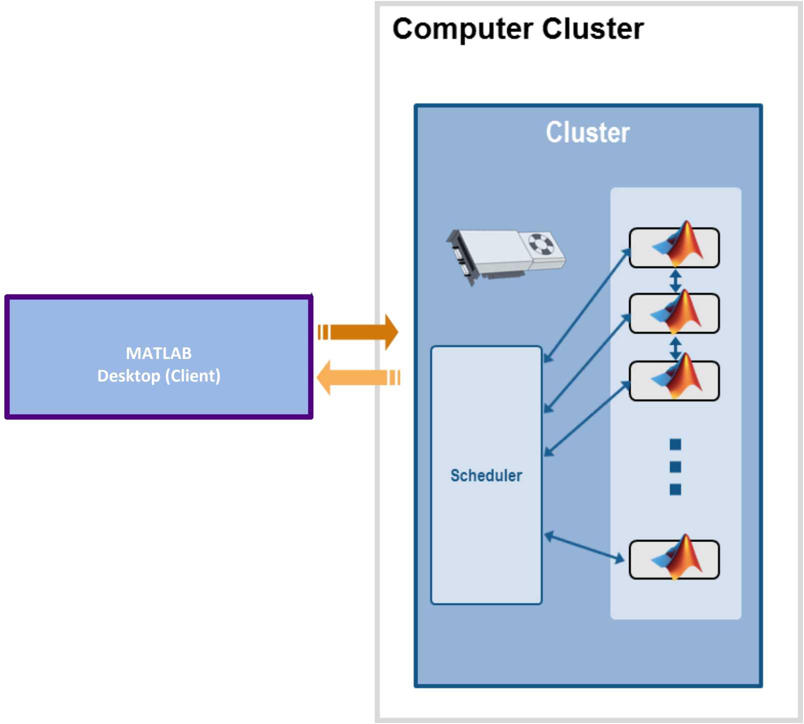
Scale to computer cluster

- Desktop interface
  - Set defaults
  - Discover clusters
  - Manage profiles
  - Monitor jobs
- Command-line API



Take Advantage of Cluster Hardware

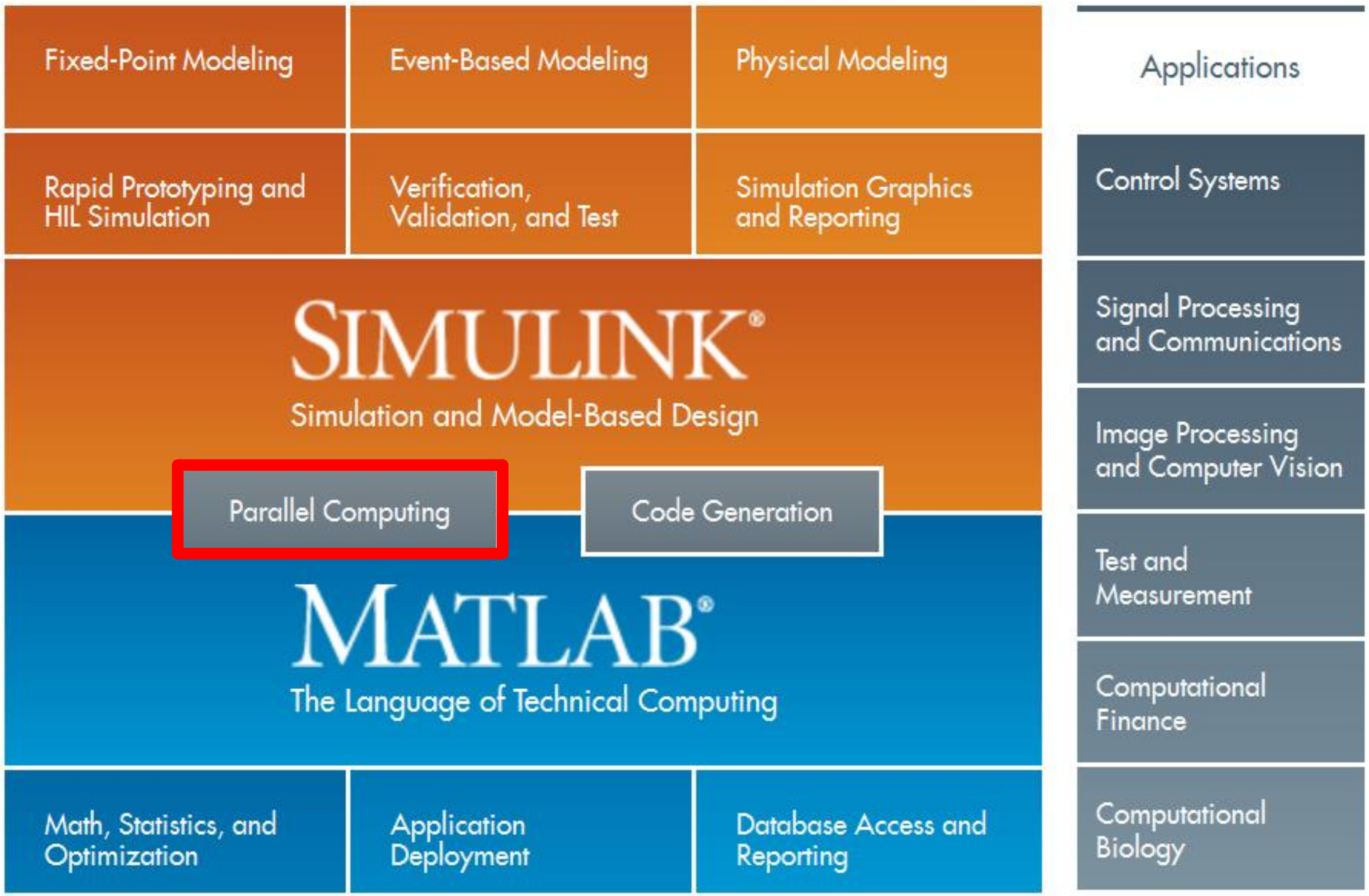
- Offload computation:
  - Free up desktop
  - Access better computers
- Scale speed-up:
  - Use more cores
  - Go from hours to minutes
- Scale memory:
  - Utilize distributed arrays
  - Solve larger problems without re-coding algorithms



Agenda

- Introduction
- Introduction to Parallel Computing
- Built in Parallel Support
- Composite Arrays and Batch Processing
- Using GPUs
- Built in GPU Support
- GPU Arrays
- Distributed GPU Computing
- Scaling Up to a Cluster
- Concluding Remarks

MathWorks Product Overview



Learn More

Online Resources

- www.mathworks.com
- www.optinum.co.za

Other services offered by OPTI-NUM solutions

- Technical Support
- Training
- Consulting
- Private Seminars

Online Resources - MathWorks

- Create MathWorks account
- Manage your account
- Obtain product updates

- View and report bugs
- Documentation
- Downloads

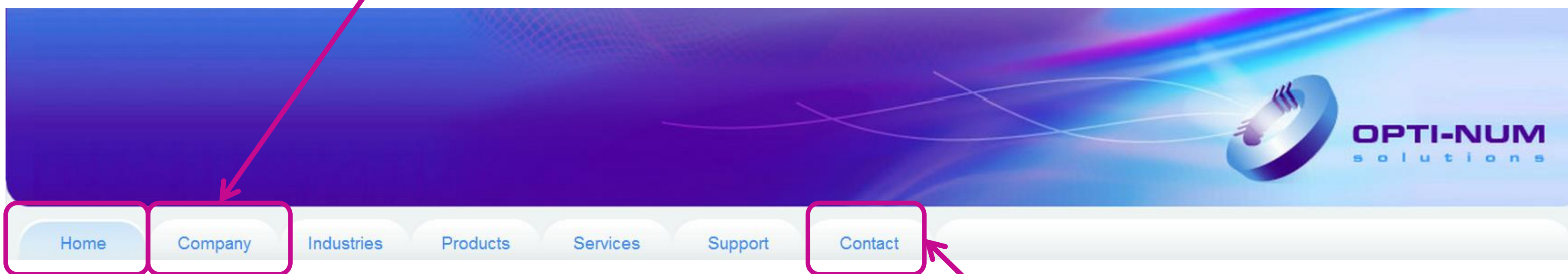
- File exchange
- Newsgroup
- Blogs

- Webinars
- Newsletters

www.mathworks.com

Online Resources – OPTI-NUM solutions

- Training schedule
- Upcoming seminars and events



- Upcoming webinars
- Latest news stories

- Submit sales or technical support queries
- Contact the consulting team

www.optinum.co.za

Technical Support

Part of Software Maintenance Service

Assistance from Applications Engineers with Bachelors degree or higher

Over 300 queries resolved in 2012

Queries can be submitted:

- By emailing support@optinum.co.za
- On the OPTI-NUM solutions website
- On the MathWorks website

Training

Public training

At our offices in Hyde Park Corner

On-site Training

Groups of four or more

For more information email sales@optinum.co.za

Introductory Courses:

- MATLAB Fundamentals (General, Aerospace and Defence, Finance, Life Sciences)
- Simulink for System and Algorithm Modelling

Intermediate Courses:

- Image Processing with MATLAB
- MATLAB Based Optimization Techniques
- Statistical Methods in MATLAB
- MATLAB for Data Processing and Visualization
- MATLAB for Building Graphical User Interfaces

Advanced Courses:

- Advanced MATLAB Programming Techniques
- Model Management and Verification in Simulink
- Stateflow for Logic Driven System Modelling
- Integrating Code with Simulink
- MATLAB and Simulink for Control Design Acceleration

Consulting Services

Provided by a team of experienced engineers

Projects range from ramp-up to full system implementation

For more information, email consulting@optinum.co.za

Questions?

What Next?

- View some webinars:
  - Parallel Computing with MATLAB on Multicore Desktops and GPU's
  - GPU Computing with MATLAB
- Do some reading:
  - MATLAB
  - Parallel Computing Toolbox
  - MATLAB Distributed Computing Server
- Related Blogs:
  - Loren on the Art of MATLAB – Parallel Computing Entries
- Keep up do date with training, free seminars and other events being hosted by OPTI-NUM solutions here

Seminar Evaluation

- Thank you for attending the workshop
- Please take a few moments to fill out the evaluation form
  - Would you like us to send you more information on the material covered today?
  - Would you like a trial of the products used?

OPTI-NUM solutions: Data Analysis with MATLAB for Excel Users

Location: **Protea Hotel Wanderers** Date: **16 May 2012**

Thank you for taking the time to join us for today's event.
Please take a moment to complete this evaluation form.

Please rate today's presentation	1 = Lowest, 5 = Highest	Were you aware of these capabilities prior to today?	Could you use these capabilities in your work?
Data Analysis with MATLAB and Excel	Content: 1 2 3 4 5 Presenter: 1 2 3 4 5	☐ Yes ☐ No	☐ Yes ☐ No
Beyond Data Analysis	Content: 1 2 3 4 5 Presenter: 1 2 3 4 5	☐ Yes ☐ No	☐ Yes ☐ No

Do you currently use MATLAB?	☐ Yes ☐ No	Do you currently use Simulink?	☐ Yes ☐ No

Overall, did the seminar content meet your expectations? Please describe.	☐ Yes ☐ No
Please describe your current mission, project or research work.	
Are you using MATLAB and/or Simulink on your current project? If yes, please explain.	☐ Yes ☐ No
Would you like us to follow-up with you about product, pricing or license questions?	☐ Yes ☐ No
Would you like us to follow-up with you about public or on-site training classes?	☐ Yes ☐ No
Would you be interested in evaluating any of the products for an upcoming or current project? If yes, please explain.	☐ Yes ☐ No
Additional comments/suggestions:	

Title: (eg. Mr, Mrs)		First:		Last:	
Company:				Department:	
Job Title:				Phone Number:	
Postal Address:				E-Mail Address:	
Town:				Post Code:	
Are you a student?	☐ No	☐ Postgraduate		☐ Undergraduate	

Lund University Develops an Artificial Neural Network for Matching Heart Transplant Donors with Recipients

Challenge

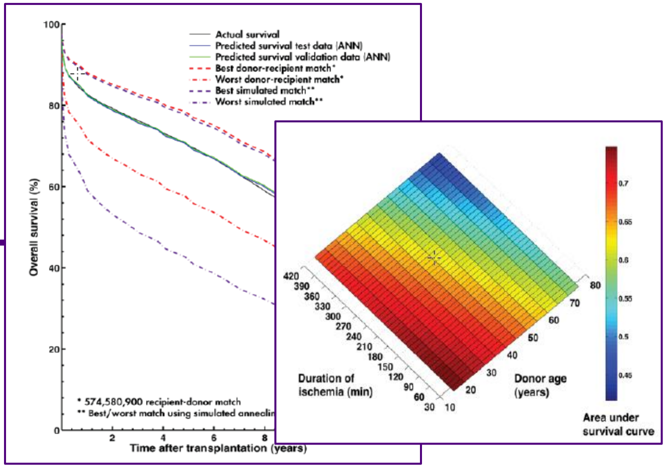
Improve long-term survival rates for heart transplant recipients by identifying optimal recipient and donor matches

Solution

Use MathWorks tools to develop a predictive artificial neural network model and simulate thousands of risk-profile combinations on a 56-processor computing cluster

Results

- Prospective five-year survival rate raised by up to 10%
- Network training time reduced by more than two-thirds
- Simulation time cut from weeks to days



Plots showing actual and predicted survival, best and worst donor-recipient match, best and worst simulated match (left); and survival rate by duration of ischemia and donor age (right).

"I spend a lot of time in the clinic, and don't have the time or the technical expertise to learn, configure, and maintain software. MATLAB makes it easy for physicians like me to get work done and produce meaningful results."

Dr. Johan Nilsson
Skåne University Hospital
Lund University