Speeding up Simulink

Murali Yeddanapudi

© 2017 The MathWorks, Inc.

Agenda

- Typical use cases
- Accelerator mode
- Performance Advisor
- Fast Restart and parsim
- Incremental workflows
- Solver Profiler

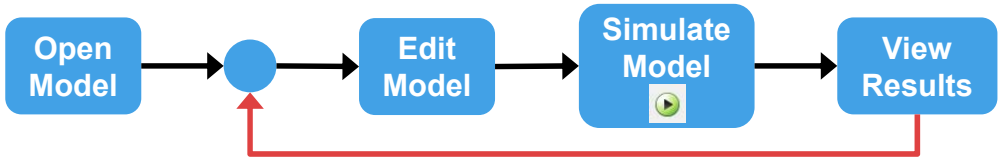
Agenda

- Typical use cases
- Accelerator mode
- Performance Advisor
- Fast Restart and parsim
- Incremental workflows
- Solver Profiler

Typical simulation use cases

Edit-Sim-Repeat
Tune-Sim-Repeat
Multi-Sim

Edit-Sim-Repeat



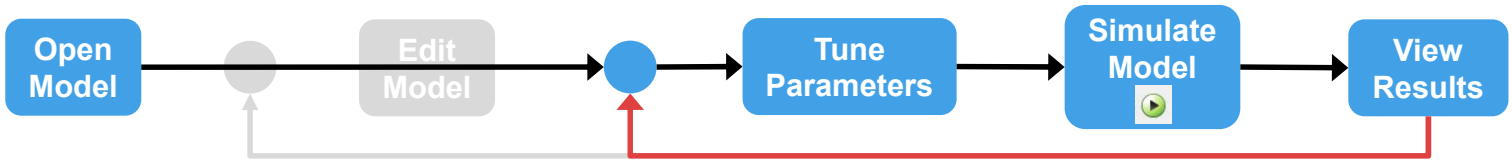
Tune-Sim-Repeat

Multi-Sim

Edit-Sim-Repeat

Tune-Sim-Repeat

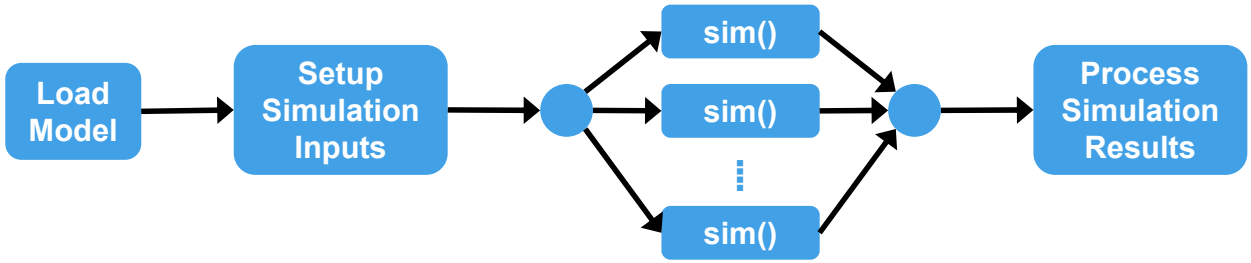
Multi-Sim



Edit-Sim-Repeat

Tune-Sim-Repeat

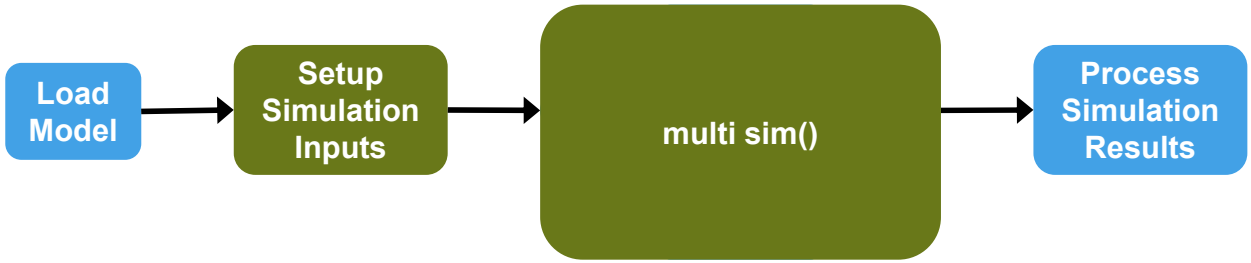
Multi-Sim



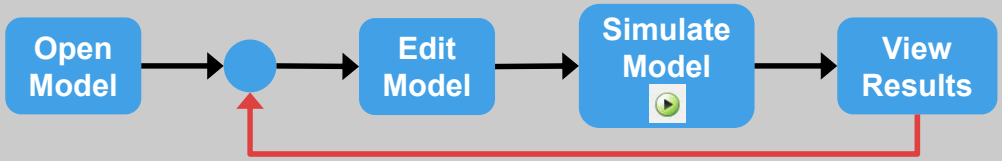
Edit-Sim-Repeat

Tune-Sim-Repeat

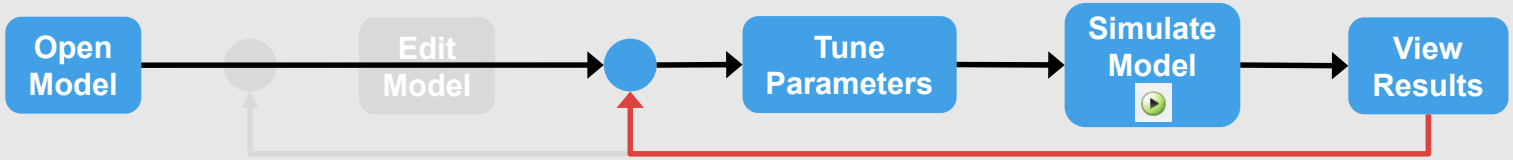
Multi-Sim



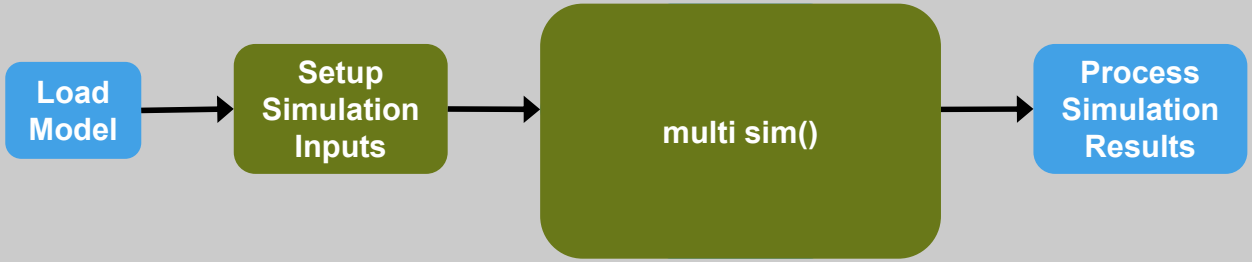
Edit-Sim-Repeat



Tune-Sim-Repeat



Multi-Sim



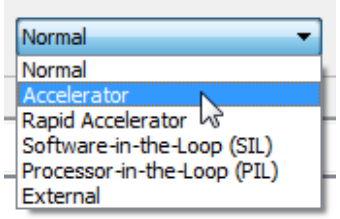
Agenda

- Typical use cases
- Accelerator mode
- Performance Advisor
- Fast Restart and parsim
- Incremental workflows
- Solver Profiler

Accelerator Mode

Why would Simulink speed up?

- JIT compiles (or generates C-code for) portions of the model
- Running compiled code has less overhead



What's the tradeoff?

- There is overhead to generate code
- Some run time diagnostics are disabled, e.g., inf/nan checking
- May not speedup all models

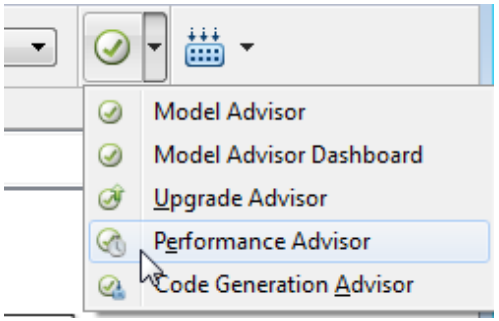
Introduced before R2006a

Help Search: how acceleration modes work

Performance Advisor

Why would Simulink speed up?

- Checks your model for speedup options
- Validates its own advice, only applies changes that:
  - give the same answer
  - and improve speed



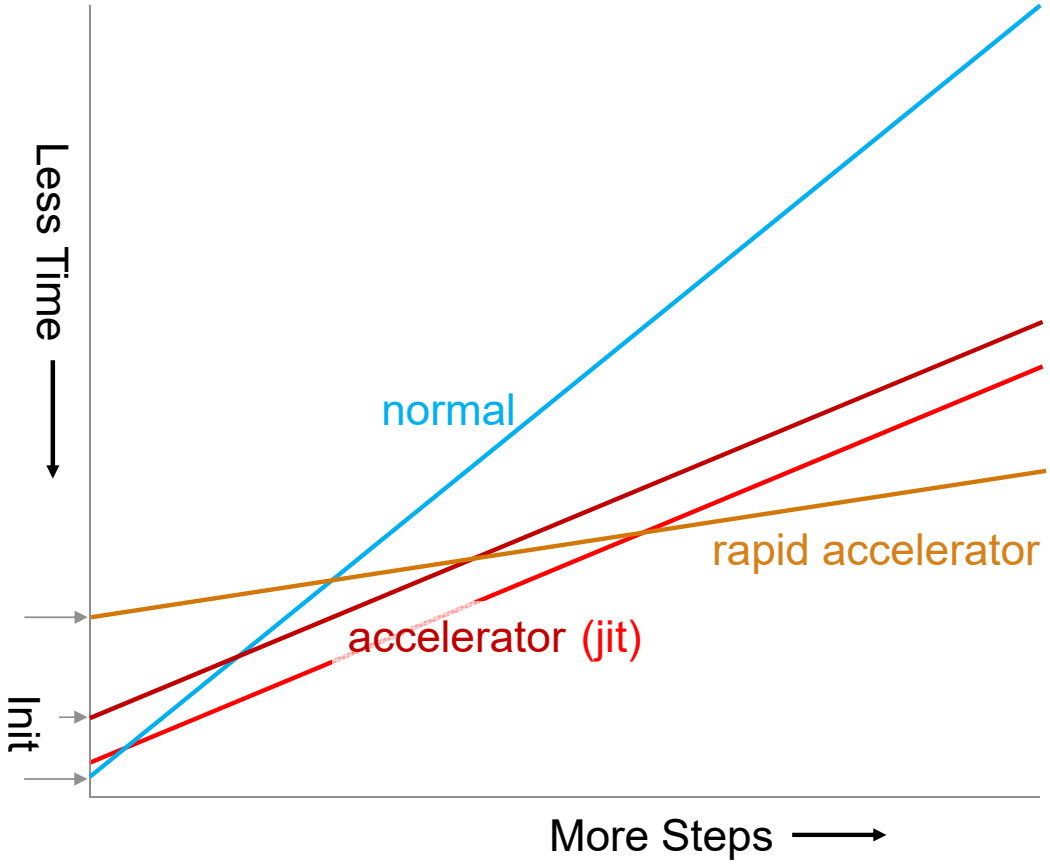
What's the tradeoff?

- Takes time run the analysis
- Not comprehensive
  - Trading off fidelity for speed is not part of performance advisor

Help Search: performance advisor

R2012b

Rough Comparison of Simulation Modes



Accelerator is faster

- Unless your simulations are short

- With JIT, accelerator is faster than normal mode in many more cases

Rapid-accelerator has the least per-step overhead but the most init overhead

Just-In-Time Accelerator Mode
Introduced in R2016b

Questions

Agenda

- Typical use cases
- Accelerator mode
- Performance Advisor
- Fast Restart and parsim
- Incremental workflows
- Solver Profiler

Fast Restart

Why would Simulink speed up?

- Avoids recompile between simulation runs
- Works with Accelerator mode

What's the tradeoff?

- Cannot edit the model when in fast restart mode

Help Search: fast restart

R2014b

parsim

R2017b
R2017a

Why would Simulink speed up?

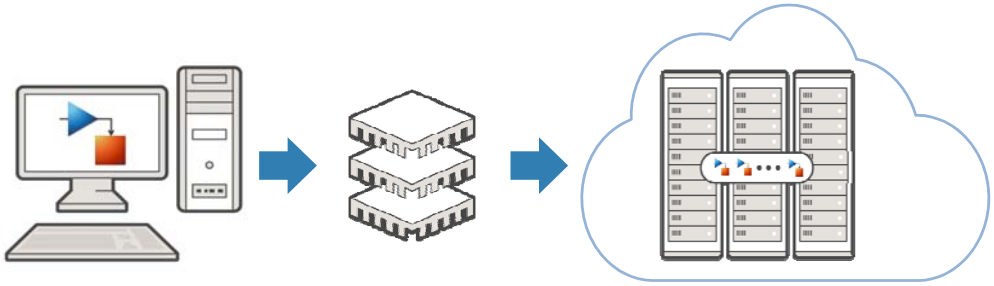
- Runs simulations in parallel using MATLAB Parallel Computing
- Parallelization details are automatically handled
  - if your model works with **`sim`** ...

    … it works with **`parsim`**

What's the tradeoff?

- Overhead of setting up parallel pool
- Overhead of starting simulations on the workers
- Needs scripting in MATLAB

```
for i = 10000:-1:1
    in(i) = Simulink.SimulationInput('my_model');
    in(i) = in(i).setVariable('my_var', i);
end
out = parsim(in);
```

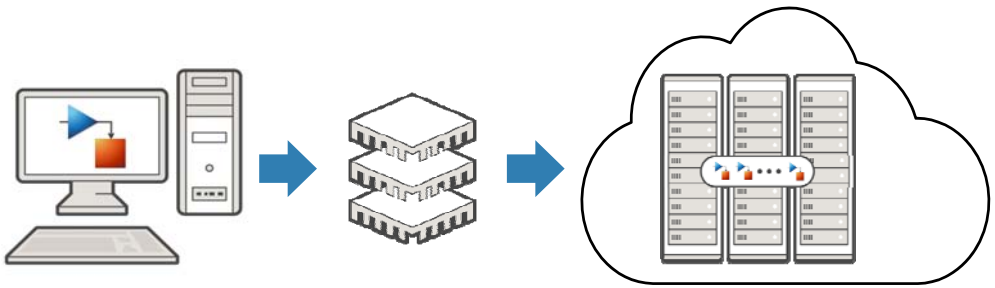


Desktop Multicore Cluster

Help Search: parsim

`parsim` : Benefits

```
for i = 10000:-1:1
    in(i) = Simulink.SimulationInput('my_model');
    in(i) = in(i).setVariable('my_var', i);
end
out = parsim(in);
```



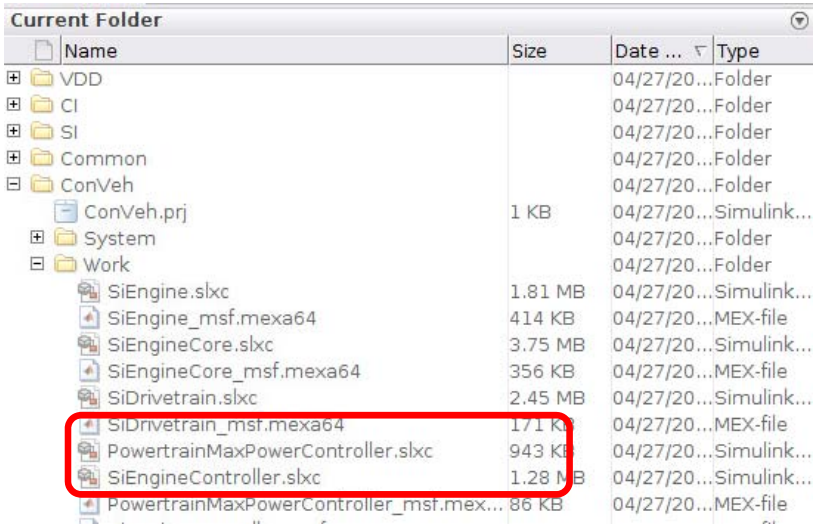
Desktop Multicore Cluster

parsim manages the details of running parallel simulations

… so you can focus on the design tasks

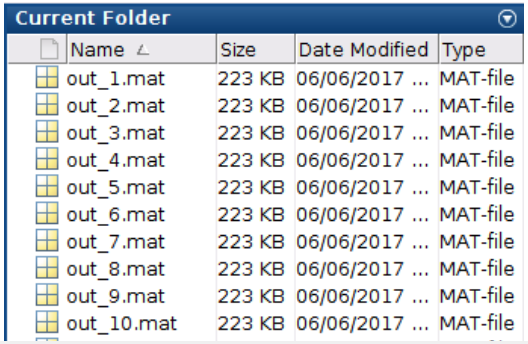
parsim: automates book-keeping details (1)

- Handles cross platform details
  - Use parsim from a Windows desktop to run simulations on Linux Cluster
- Handles model dependencies
  - MATLAB Code, Libraries, S-Functions, …
- Integrated with Simulink Cache
- Leverages model reference parallel build



parsim: automates book-keeping details (2)

- Brings back log files from the workers
  - Appends run id to make them unique

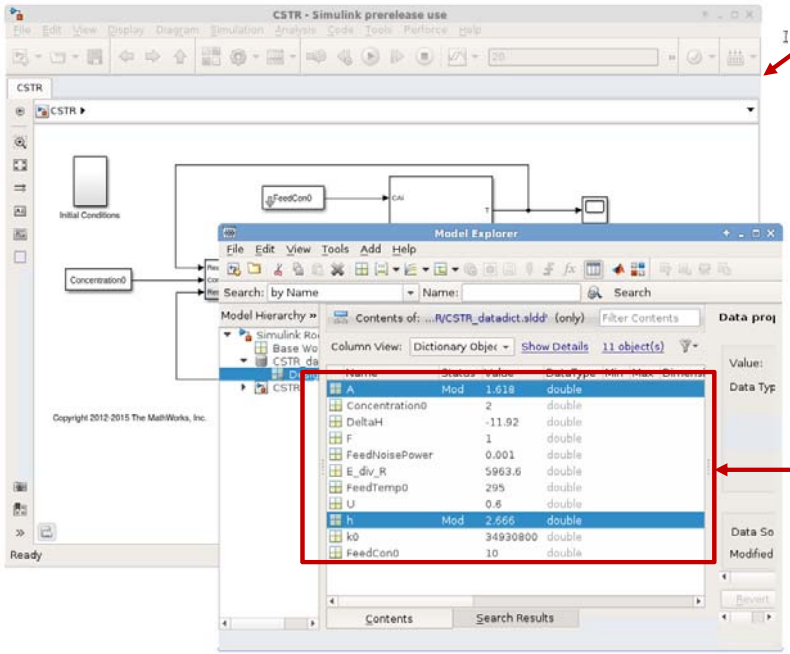


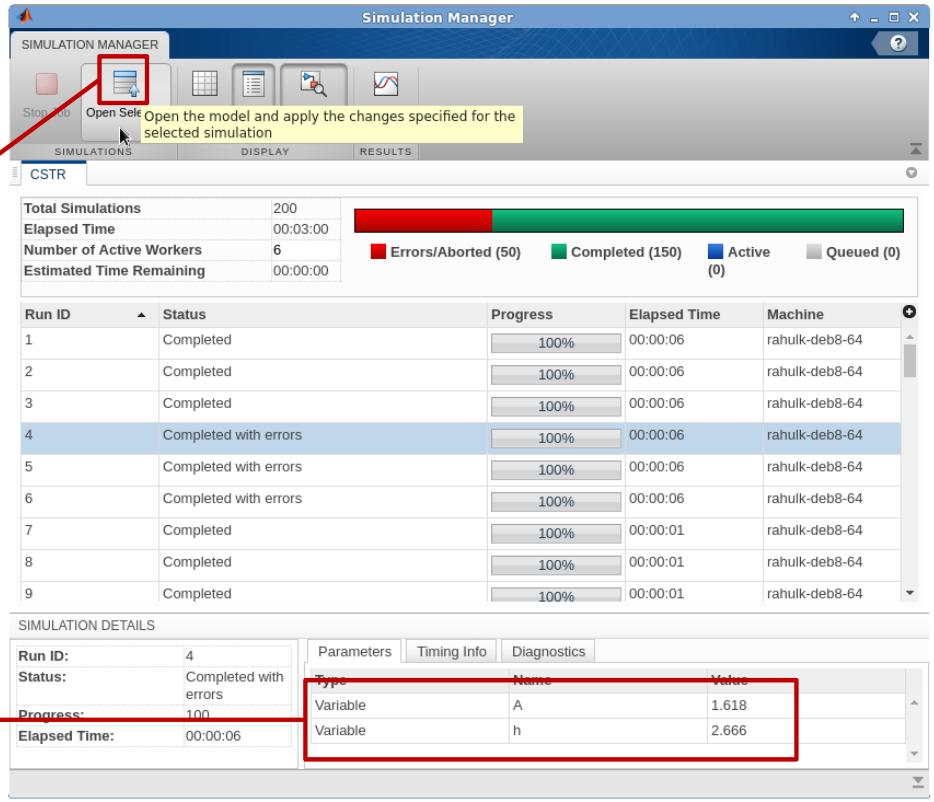

```
>> out(198)

  Simulink.SimulationOutput:

                    tout: [141565x1 double]
                 logsout: [1x1 Simulink.SimulationData.DatasetRef]

    SimulationMetadata: [1x1 Simulink.SimulationMetadata]
          ErrorMessage: [1x15267 char]
```

Automatically get references to logged files

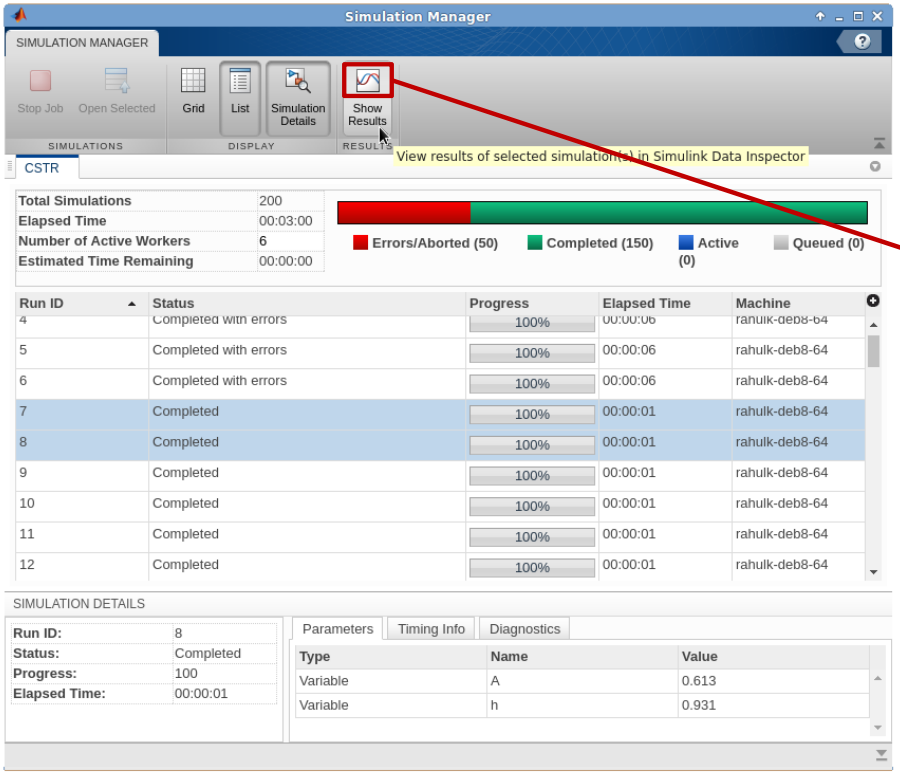
parsim: automates book-keeping details (3)

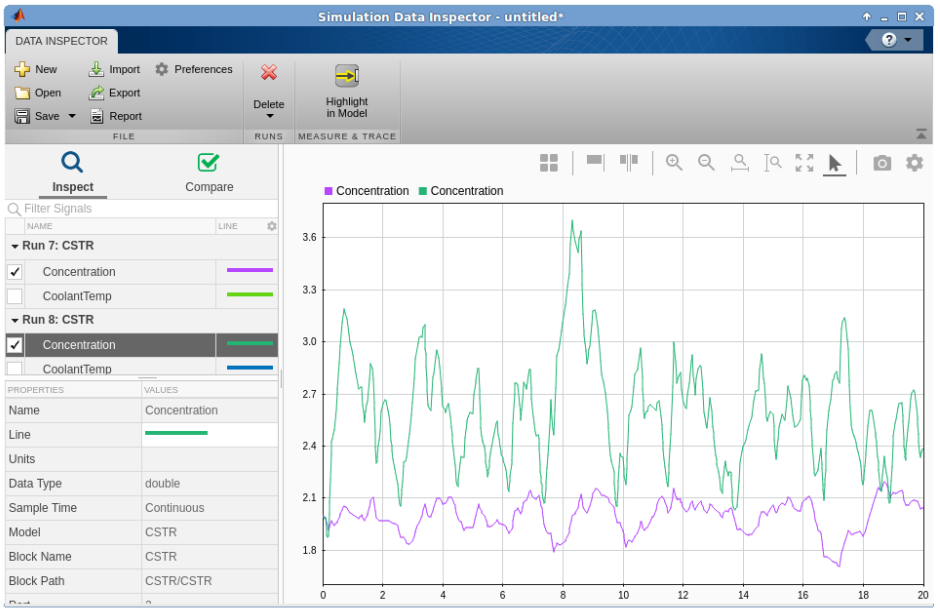
- Show progress and error diagnostics
  - Setups up model to run locally to debug





Visualizing Results





`parsim` : customization(1)

- TransferBaseWorkspaceVariables

```
outs = parsim(inps, 'TransferBaseWorkspaceVariables','on', …)
```

- UseFastRestart

```
outs = parsim(inps, 'UseFastRestart','on', …)
```

`parsim` : customization(2)

- SetupFcn

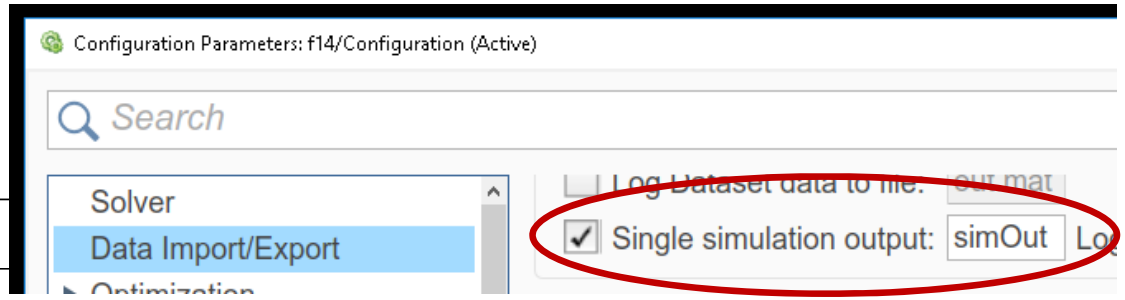
```
setupFcn = @()addpath('myProjectDir')
outs = parsim(inps, 'SetupFcn',setupFcn, …)
```

- CleanupFcn

```
cleanupFcn = @()rmpath('myProjectDir')
outs = parsim(inps, 'CleanupFcn',cleanupFcn, …)
```

SimulationOutput object

```
simOut = sim('model',
```




```
Trial>> simOut

  Simulink.SimulationOutput:

            ScopeData1: [1x1 Simulink.SimulationData.Dataset]
            ScopeData2: [1x1 struct]
                  tout: [1353x1 double]
                  xout: [1x1 struct]
                  yout: [1x1 struct]

    SimulationMetadata: [1x1 Simulink.SimulationMetadata]
          ErrorMessage: [0x0 char]
```

- Contains all logged simulation data
- Use dot notation to access the data
- Introduced in R2009a

SimulationInput object

A SimulationInput object '**simInp**' encapsulates all input to one simulation

```
simOut = sim(simInp)
```

Array of **simInps** encapsulate all inputs to multiple simulations

```
simOuts = sim(simInps)
```

* Simulations are run *sequentially*

```
simOuts = parsim(simInps)
```

Simulations are run in **parallel** if MATLAB parallel computing tools are available, *serially* otherwise

SimulationInput Object

```
SimulationInput with properties:

         ModelName: 'sldemo_suspn_3d
    InitialState: [0x0 Simulink.Si
   ExternalInput: []
ModelParameters: [1x1 Si
BlockParameters: [0x0 Simuli
       Variables: [1x2 Sim
      PreSimFcn: []
     PostSimFcn: []
      UserString: ''
```

Specify MATLAB functions to run before and after each

Add a brief UserString describing these changes for easy reference

Change variables in base workspace, data dictionary, or model workspace

PreSimFcn

- Use PreSimFcn to offload parameter computations to parallel workers

```
for i = 10:-1:1
    in(i) = Simulink.SimulationInput(i);
    in(i).PreSimFcn = @(inp) myPreSimFcn(inp, i);
end

function simInp = myPreSimFcn(rawSimInp, runId)
    prmValue = expensiveComputation(runId);
    simInp = rawSimInp.setBlockParameter( …
      [rawSimInp.ModelName,'/my_block'], 'prmName', prmValue);
end
```

PostSimFcn

- use PostSimFcn to post-process raw simulation outputs in parallel
- reduce data returned back from workers

```
>> inps = Simulink.SimulationInput('myModel');
>> …
>> inps.PostSimFcn = @(out) myPostSimFcn(out);
>> outs = parsim(inps);
>> outs(i).result

function simOut = myPostSimFcn(rawSimOut)
    simOut.result = expensivePostProc(rawSimOut.lotsOfLogsOut);
end
```

Questions

Agenda

- Typical use cases
- Accelerator mode
- Performance Advisor
- Fast Restart and parsim
- Incremental workflows
- Solver Profiler

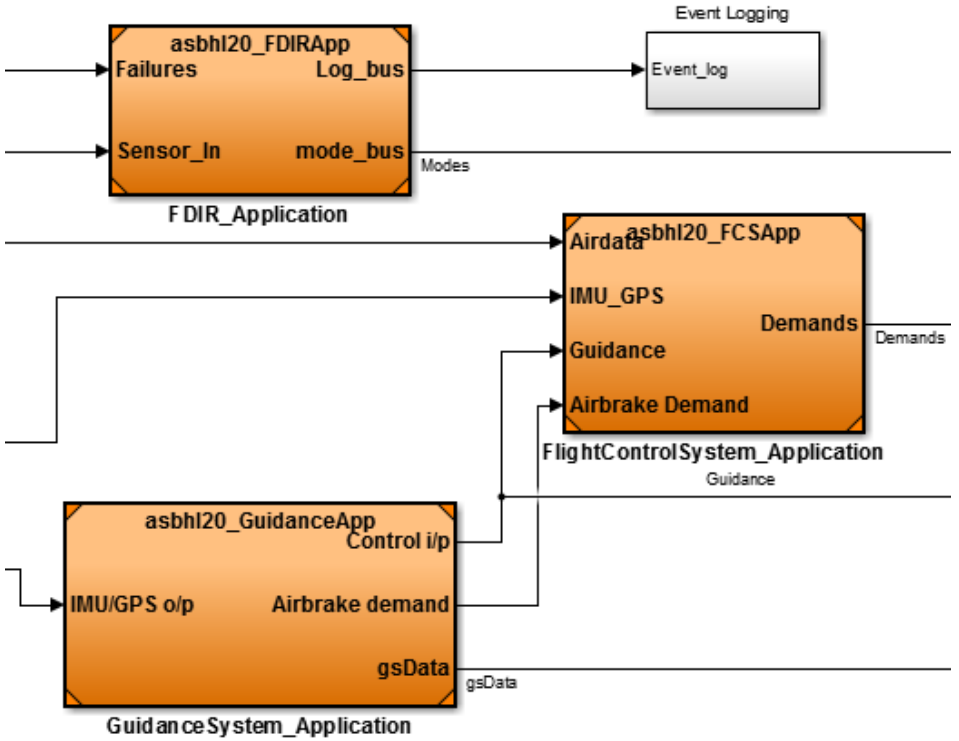
What is an incremental workflow?

Only perform an action when necessary;

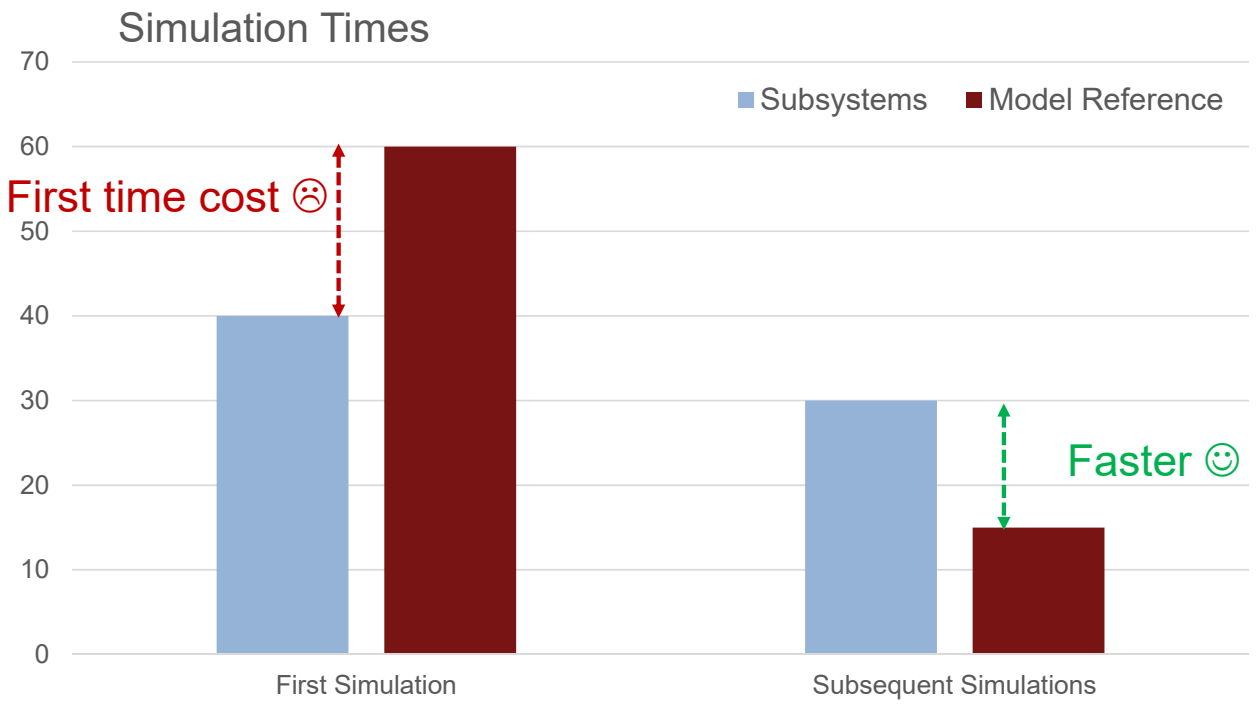
reuse and cache as much as possible

Model reference: incremental workflows

- Incremental Loading
- Incremental Update Diagram
- Incremental Code Generation
- Selective acceleration



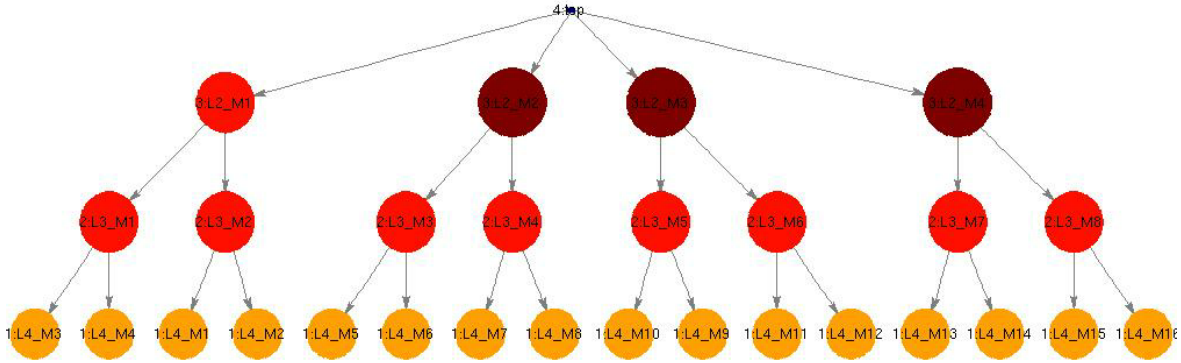
Model Reference: Performance



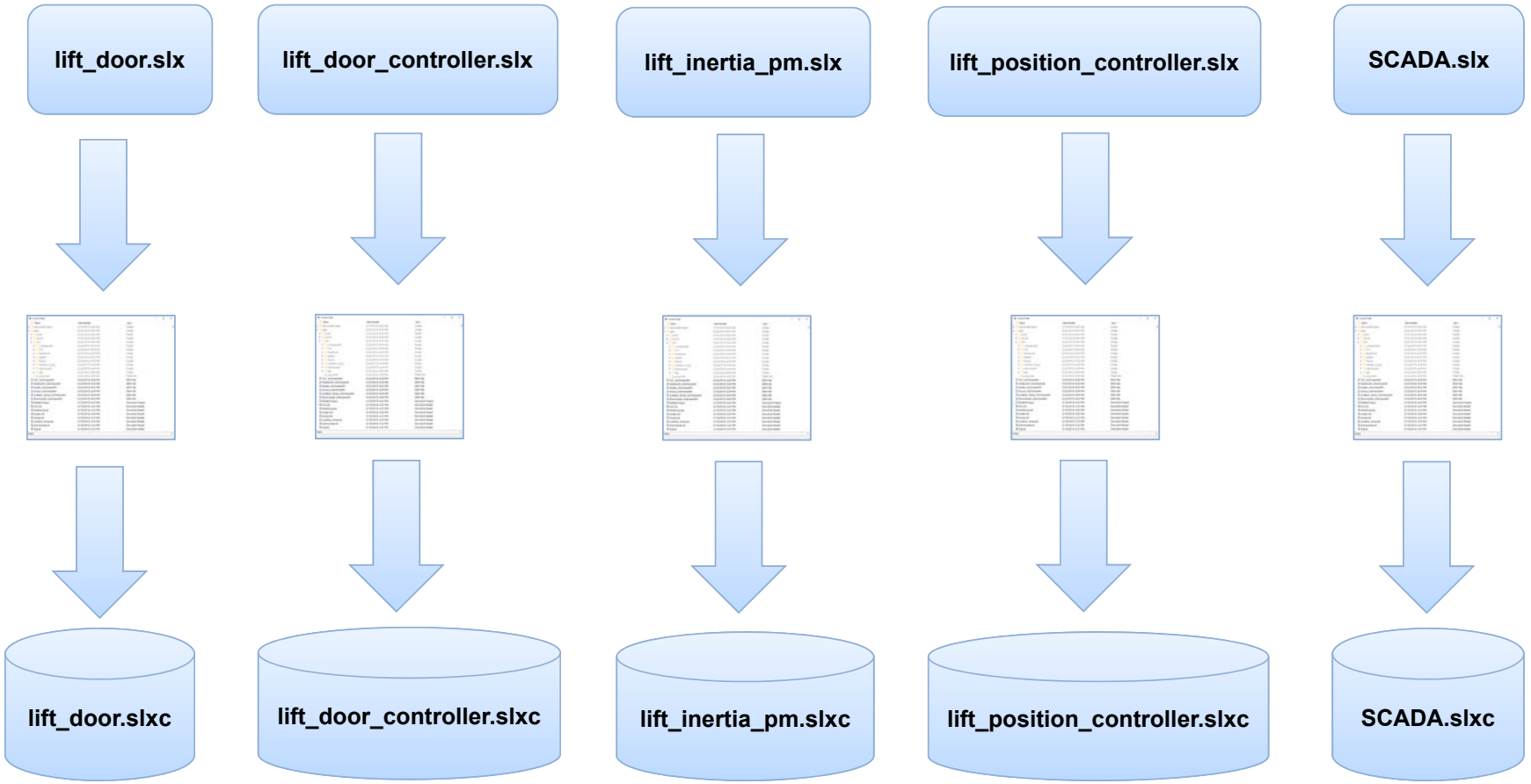
How to reduce first time cost?

Simulink Cache

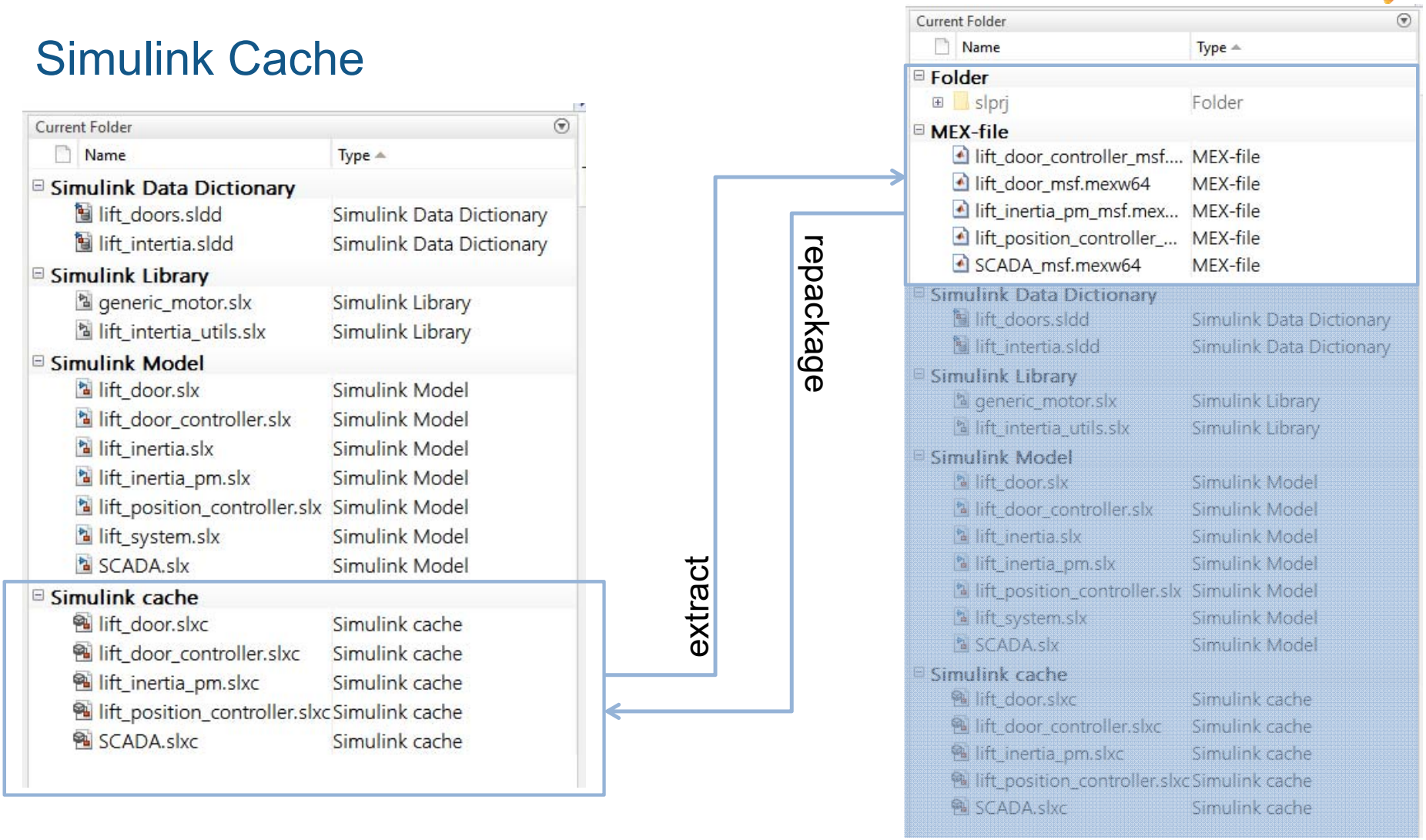
Parallel Model Reference Build



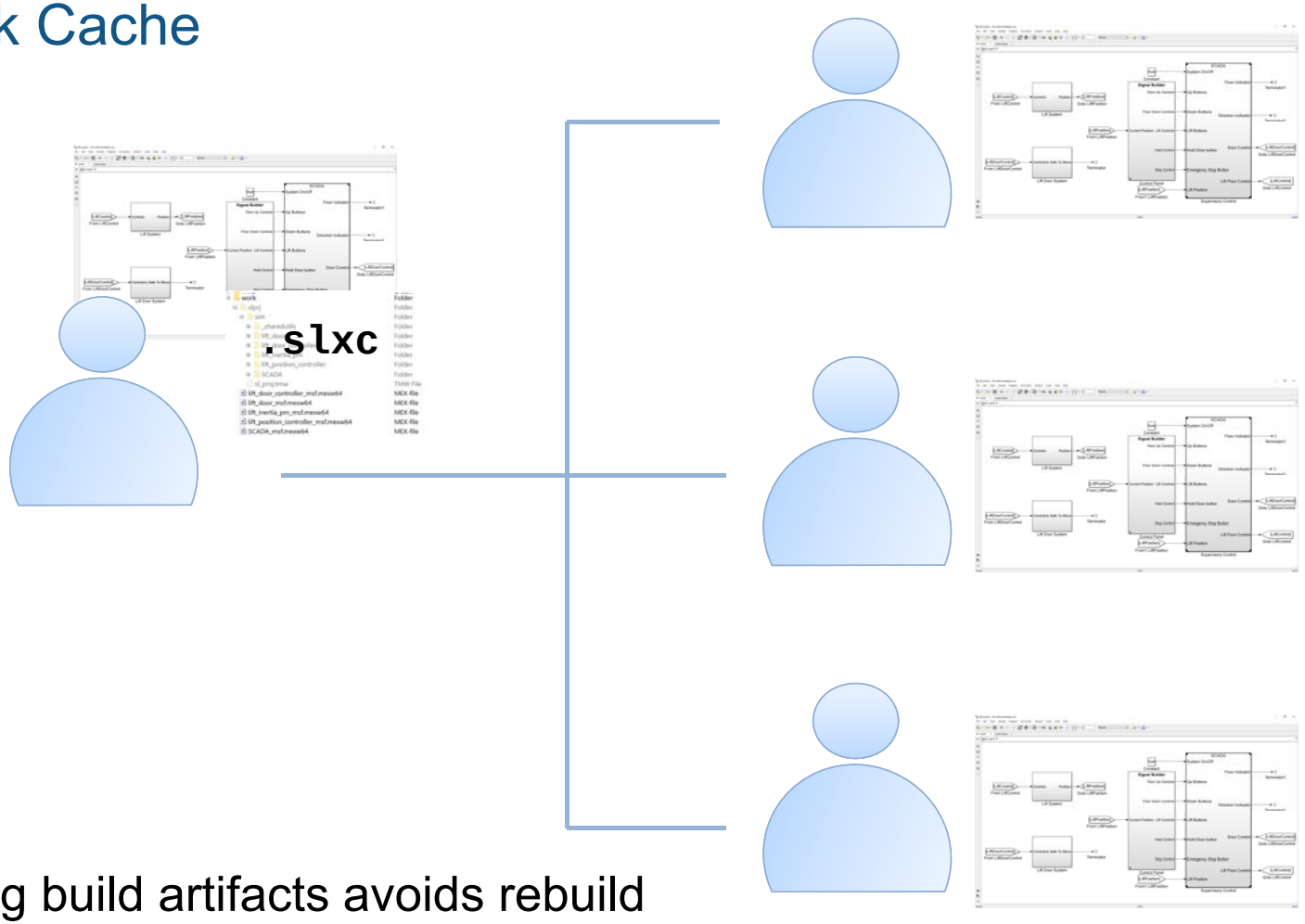
Simulink Cache



Simulink Cache



Simulinк Cache



- Sharing build artifacts avoids rebuild

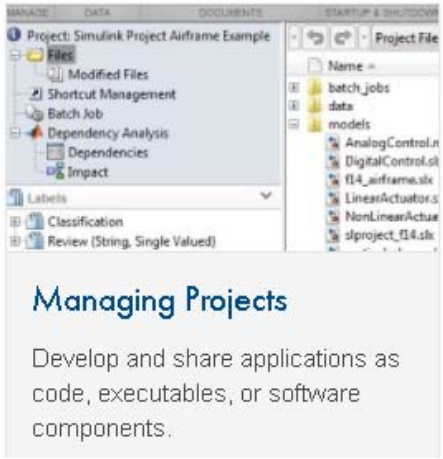
Simulink Cache

Why would Simulink speed up?

- Sharing build artifacts reduces first time cost
- Integrated into Simulink Projects and parsim

What's the tradeoff?

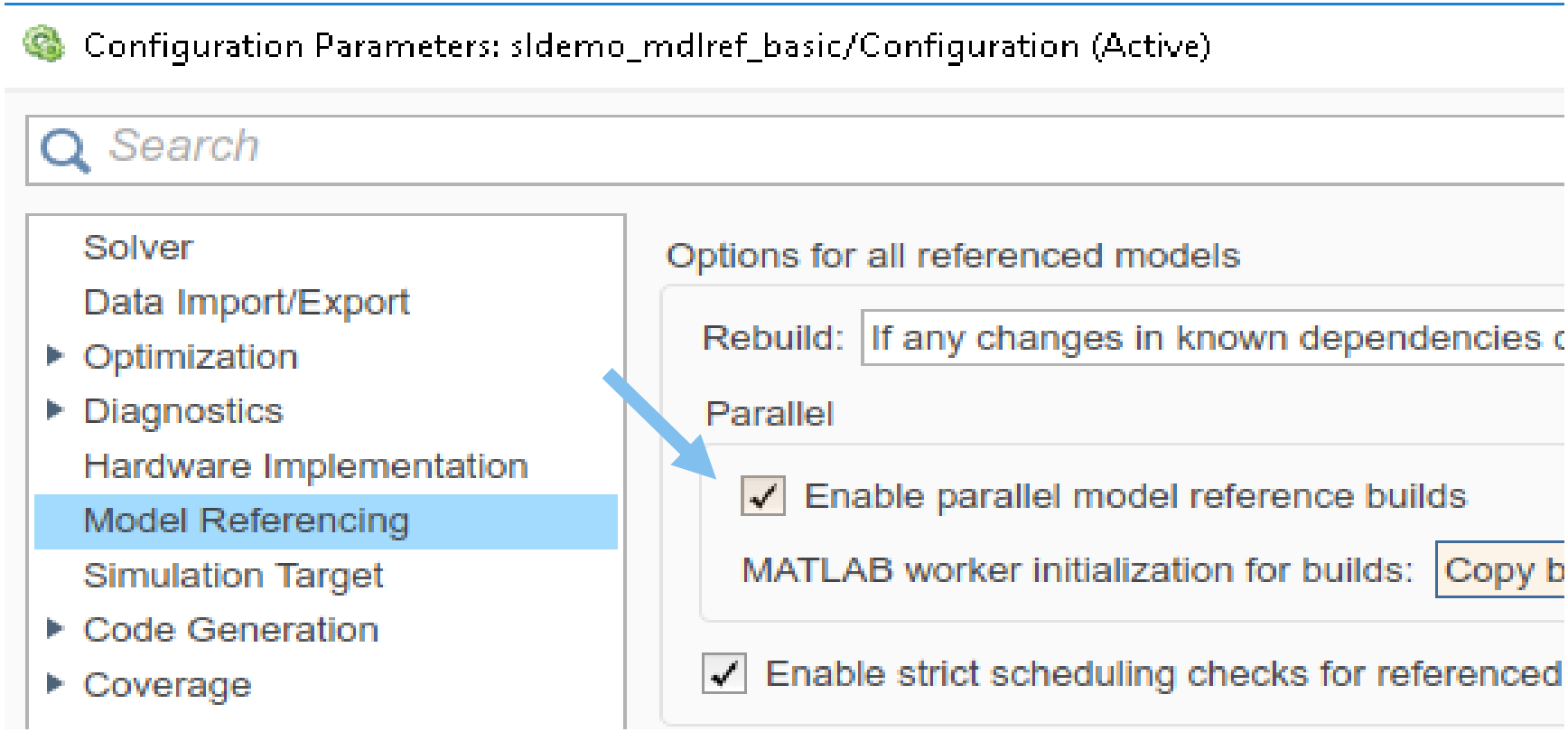
- Extra work needed to manage `.slxc` files
  - If Simulink Projects is not used



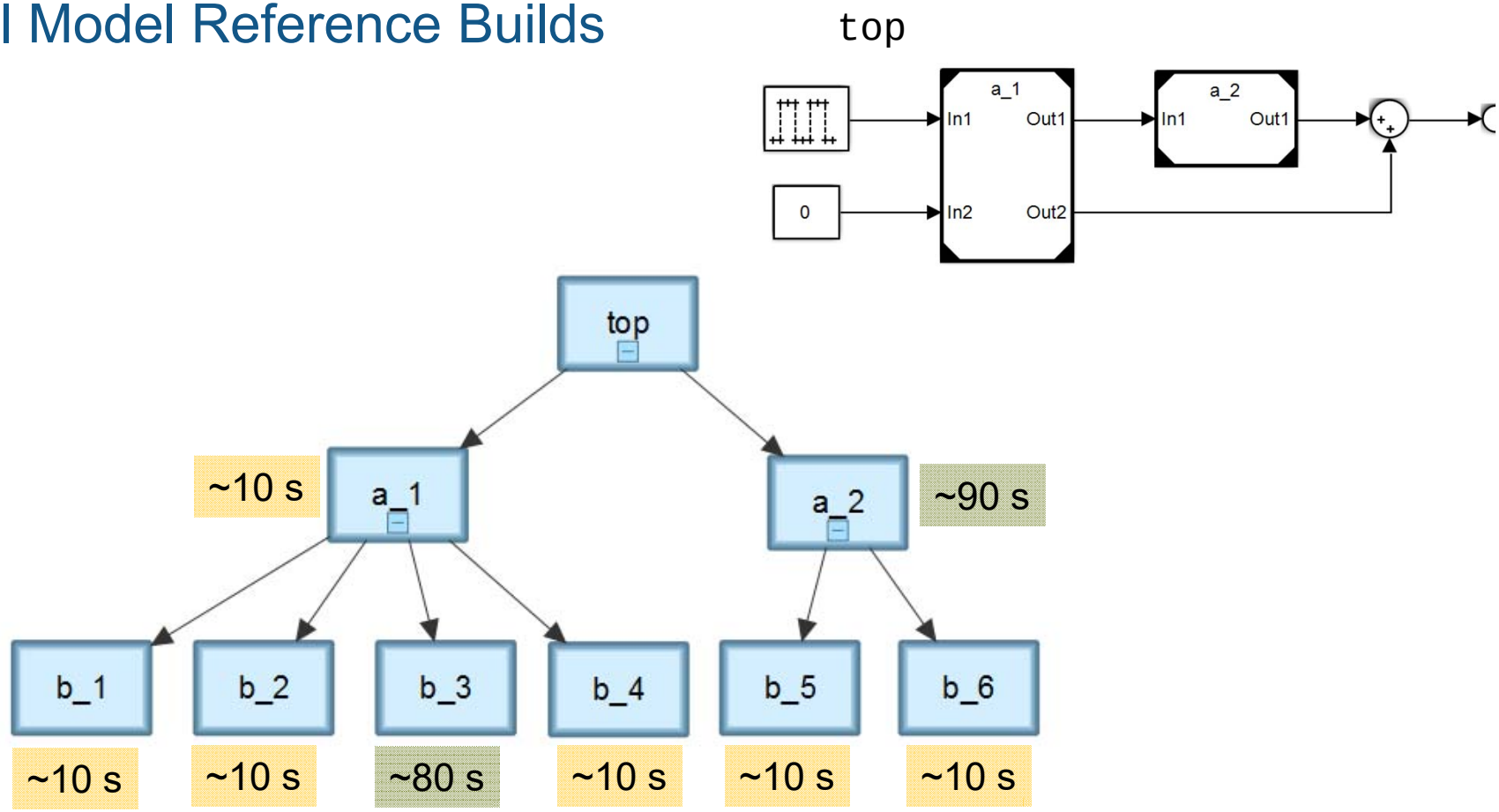
Help Search: simulink cache

R2017a

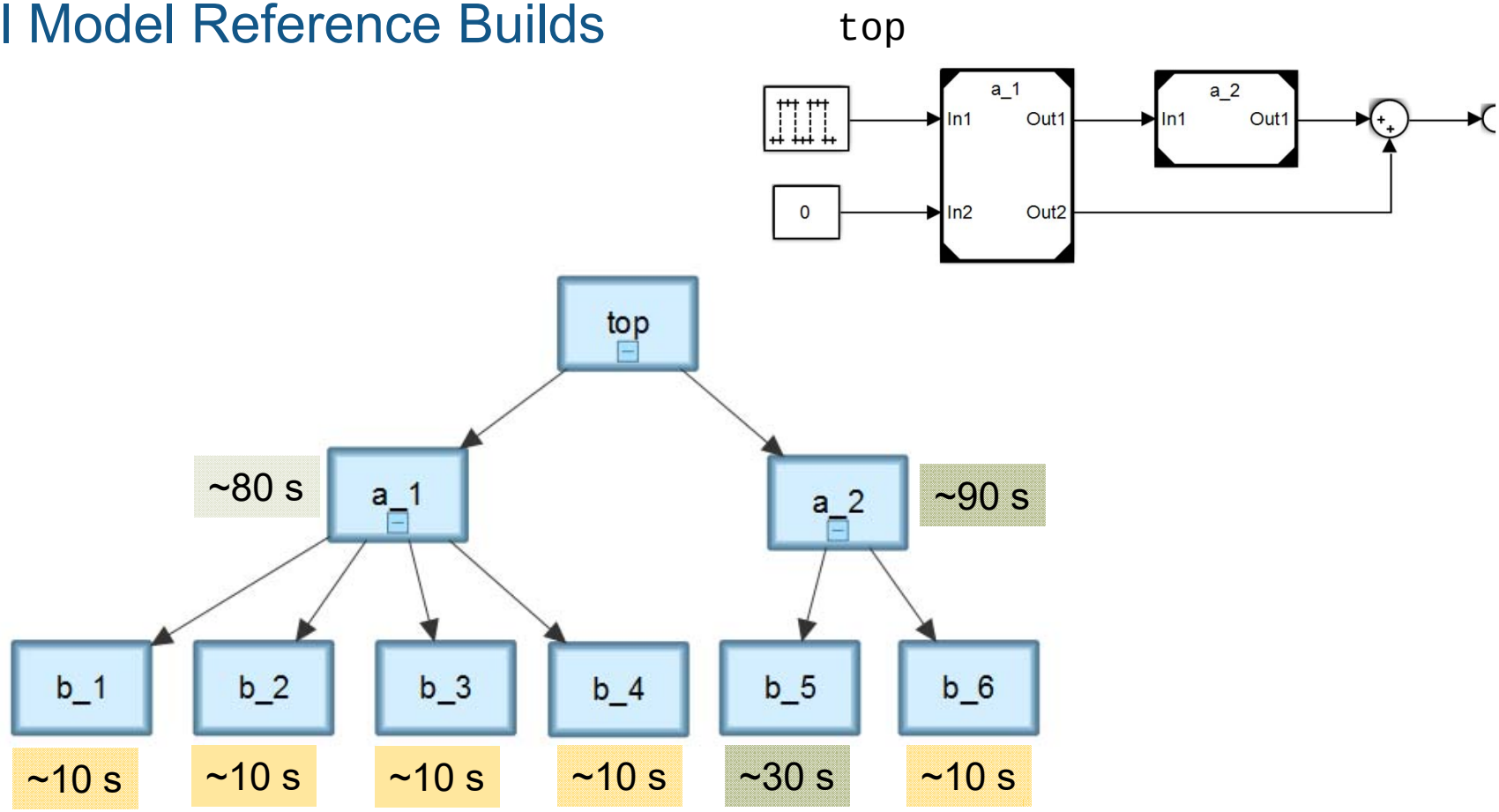
Parallel Model Reference Builds



Parallel Model Reference Builds

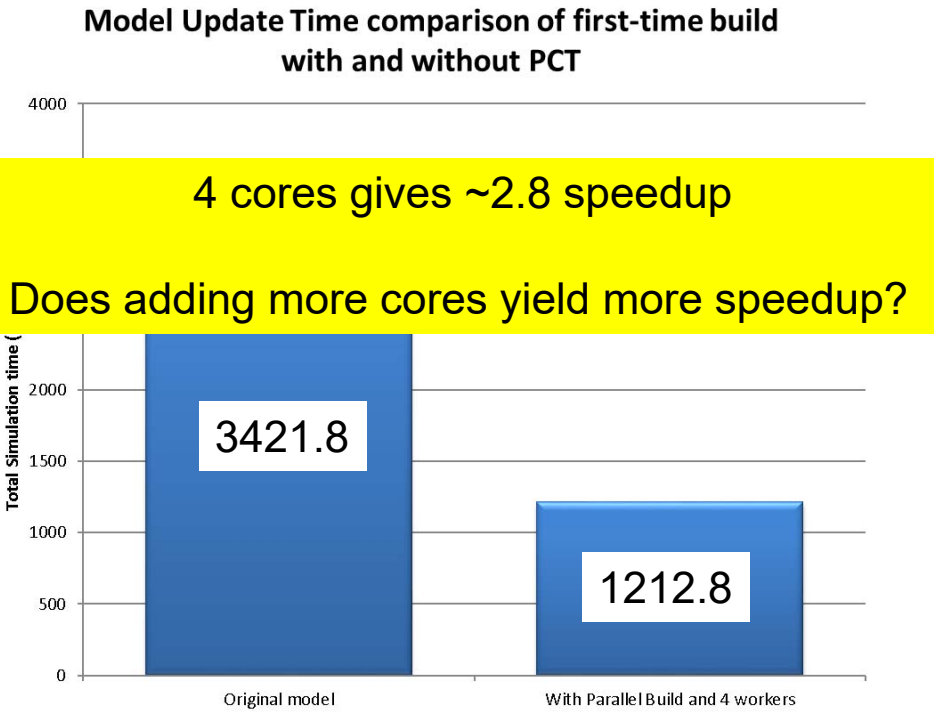


Parallel Model Reference Builds



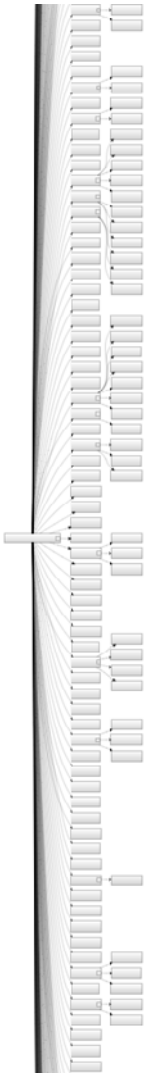
Model Reference Parallel Build User example

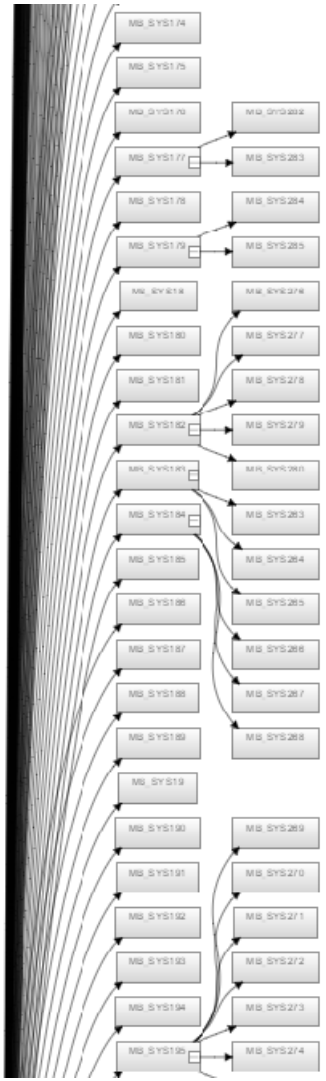
- Approximately 400 referenced models



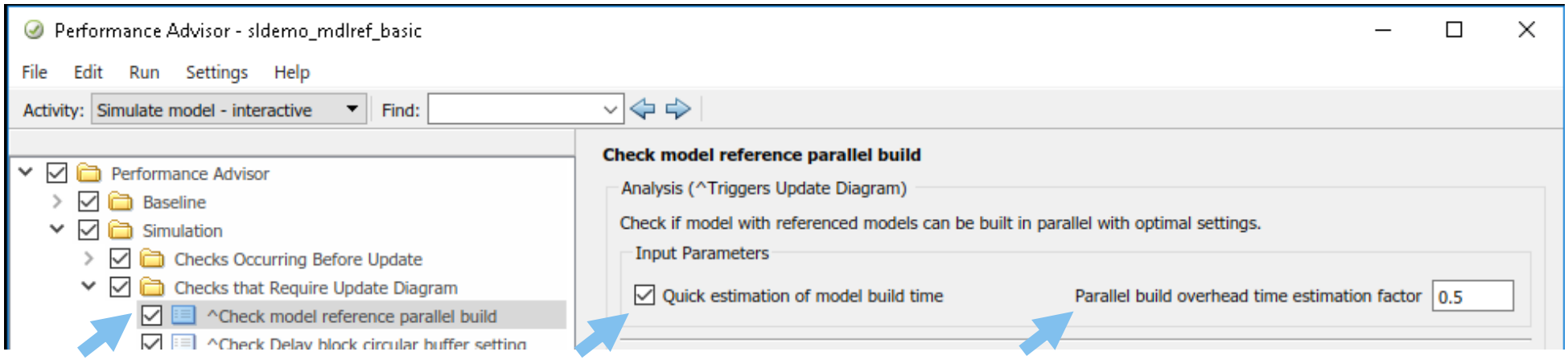
4 cores gives ~2.8 speedup

Does adding more cores yield more speedup?





Performance Advisor: Check model reference parallel build



Performance Advisor: Check model reference parallel build

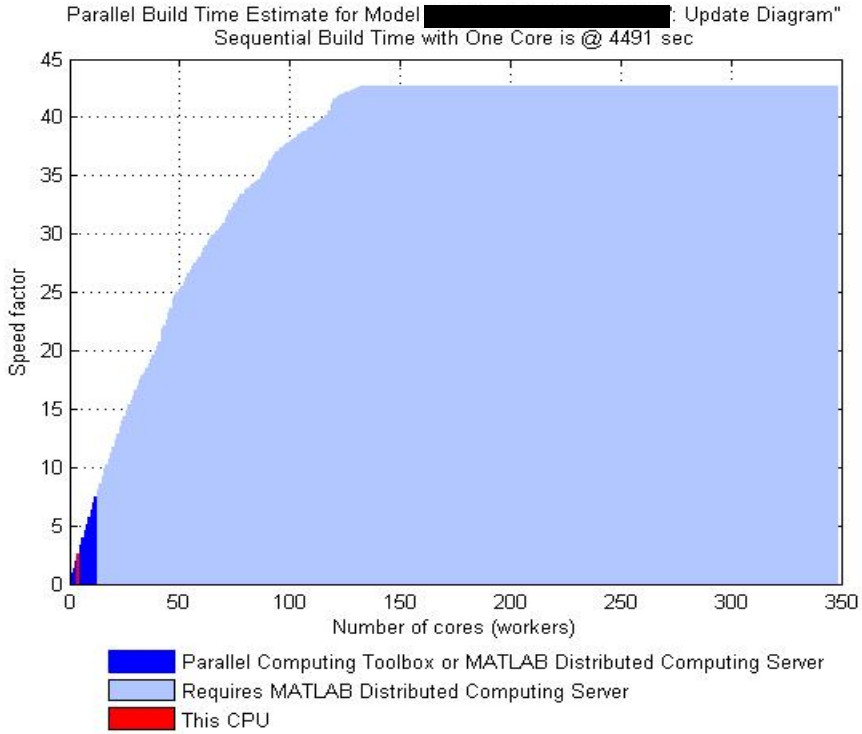
- Performance Advisor estimates the speedup with more cores
- The estimated speed up with 4 cores is ~2.6
  - Close to the measured value ~2.8
- Given ~120 cores, the estimated speed up is ~42

  => Build time goes from ~3400s to ~80s

Analysis and Advice:

Estimate build time speedup using number of 4 cores of this CPU: ---- **--2.6268x**
Estimate build time speedup using 348 workers: ---- **--42.6475x**

Estimated build times for various worker counts are as follows:



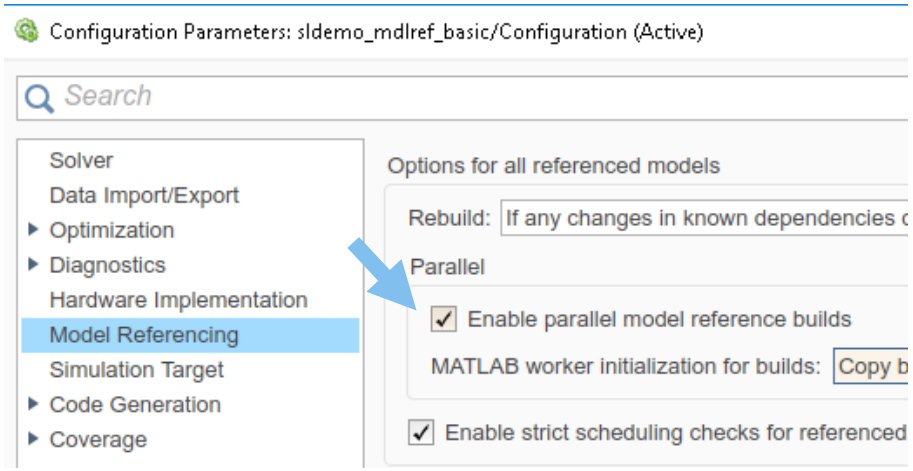
Model Reference Parallel Build

Why would Simulink speed up?

- Model reference targets are built in parallel
- Use Performance Advisor to check if your large models can benefit from this option

What's the tradeoff?

- Speedup is model dependent
- Requires MATLAB Parallel Computing



R2012b

Help Search: model reference parallel build

Questions

Agenda

- Typical use cases
- Accelerator mode
- Performance Advisor
- Fast Restart and parsim
- Incremental workflows
- Solver Profiler

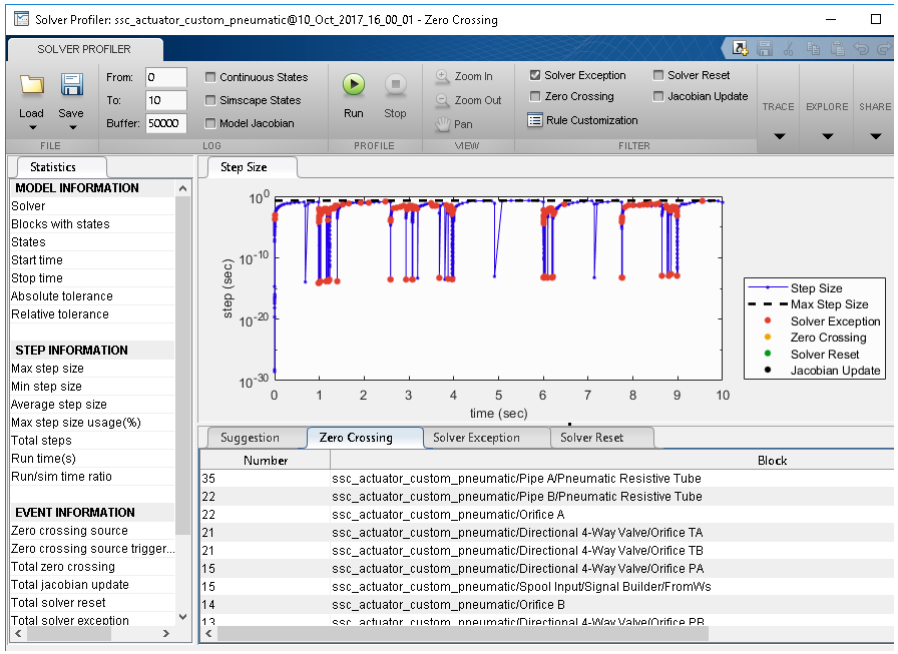
Solver Profiler

Why would Simulink speed up?

- Identifies parts of the model causing solver to slow down
  - too many resets
  - too many zero crossings etc.

What's the tradeoff?

- Profiling overhead
- Requires domain knowledge to optimally fix the issues identified by the Solver profiler.



Help Search: solver profiler

R2016a

Summary

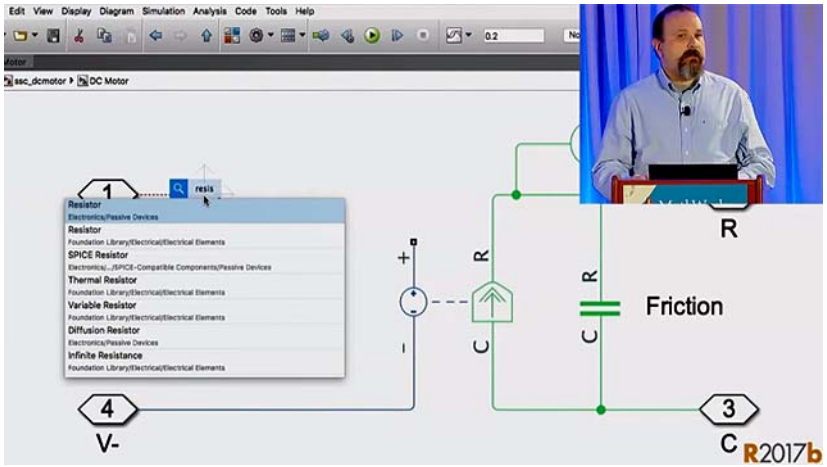
Use Case	Recommended Features
▪ Edit-Sim-Repeat	▪ Normal mode ▪ Accelerator mode
▪ Tune-Sim-Repeat	▪ Performance Advisor ▪ Fast Restart ▪ Accelerator + Fast Restart
▪ Multi-Sim	▪ parsim ▪ parsim + Accelerator + Fast Restart ▪ parsim + Rapid Accelerator + Up-To-Date-Check-Off

Smart Editing

Model Reference

Simulation Data Inspector

Solver Profiler



[Editing at the Speed of Thought with Simulink]

Learn about the latest smart editing features that have been added to Simulink to increase your modeling speed.

Thank you