Speeding up Simulink

Murali Yeddanapudi
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

Open Model → Edit Model → Tune Parameters → Simulate Model → View Results
Edit-Sim-Repeat

Tune-Sim-Repeat

Multi-Sim

Load Model ➔ Setup Simulation Inputs ➔ sim() ➔ sim() ➔ ... ➔ Process Simulation Results
Edit-Sim-Repeat

Tune-Sim-Repeat

Multi-Sim

Load Model → Setup Simulation Inputs → multi sim() → Process Simulation Results
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 speed up 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
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*
parsim

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

Help Search: parsim

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

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

  ```matlab
  outs = parsim(inps, 'TransferBaseWorkspaceVariables', 'on', ...)
  ```

- UseFastRestart

  ```matlab
  outs = parsim(inps, 'UseFastRestart', 'on', ...)
  ```
parsim : customization(2)

- SetupFcn

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

- CleanupFcn

```matlab
cleanupFcn = @( )rmpath('myProjectDir')
outs = parsim(inps, 'CleanupFcn',cleanupFcn, ...)
```
SimulationOutput object

```matlab
simOut = sim('model', 'ReturnWorkspaceOutputs','on', ...)
```

- 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

\[
\text{simOut} = \text{sim}(\text{simInp})
\]

Array of `simInps` encapsulate all inputs to multiple simulations

\[
\text{simOuts} = \text{sim}(\text{simInps})
\]

* Simulations are run *sequentially*

\[
\text{simOuts} = \text{parsim}(\text{simInps})
\]

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

**SimulationInput** with properties:

- ModelName: 'sldemo_suspn_3dof'
- InitialState: [0x0 Simulink.SimState.ModelSimState]
- ExternalInput: []
- ModelParameters: [1x1 Simulink.Simulation.ModelParameter]
- BlockParameters: [0x0 Simulink.Simulation.BlockParameter]
- Variables: [1x2 Simulink.Simulation.Variable]
- PreSimFcn: []
- PostSimFcn: []
- UserString: ''

Specify MATLAB functions to run before and after each simulation. Add a brief UserString describing these changes for easy reference. Change model or block parameters, change variables in base workspace, data dictionary, or model workspace.
PreSimFcn

- Use PreSimFcn to offload parameter computations to parallel workers

```matlab
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

```MATLAB
>> 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

Simulation Times

First time cost 😞

Faster 😊

First Simulation

Subsequent Simulations
How to reduce first time cost?

Simulink Cache

Parallel Model Reference Build
Simulink Cache

- **lift_door.slx**
- **lift_door_controller.slx**
- **lift_inertia_pm.slx**
- **lift_position_controller.slx**
- **SCADA.slx**

**Cache Levels**

- **lift_door.slxc**
- **lift_door_controller.slxc**
- **lift_inertia_pm.slxc**
- **lift_position_controller.slxc**
- **SCADA.slxc**
Simulink Cache

- **Simulink Data Dictionary**
  - lift_doors.sldd
  - lift_intertia.sldd

- **Simulink Library**
  - generic_motor.slx
  - lift_intertia_utils.slx

- **Simulink Model**
  - lift_door.slx
  - lift_door_controller.slx
  - lift_inertia.slx
  - lift_inertia_pm.slx
  - lift_position_controller.slx
  - lift_system.slx
  - SCADA.slx

- **Simulink cache**
  - lift_door.slx
  - lift_door_controller.slx
  - lift_inertia_pm.slx
  - lift_position_controller.slx
  - SCADA.slx

- **MEX-file**
  - lift_door_controller_msf.mex...
  - lift_door_msf.mexw64
  - lift_inertia_pm_msf.mex...
  - lift_position_controller_msf.mex...
  - SCADA_msf.mexw64

Repackage

Extract
Simulink 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
Parallel Model Reference Builds

Configuration Parameters: sldemo.mdlref_basic/Configuration (Active)

- Solver
- Data Import/Export
- Optimization
- Diagnostics
- Hardware Implementation
- Model Referencing
- Simulation Target
- Code Generation
- Coverage

Options for all referenced models

- Rebuild: If any changes in known dependencies or build settings
- Parallel
  - Enable parallel model reference builds
- MATLAB worker initialization for builds: Copy...
- Enable strict scheduling checks for referenced models
Parallel Model Reference Builds

```
<table>
<thead>
<tr>
<th></th>
<th>a_1</th>
<th></th>
<th>a_2</th>
<th></th>
<th>top</th>
</tr>
</thead>
<tbody>
<tr>
<td>b_1</td>
<td>~10 s</td>
<td>b_2</td>
<td>~10 s</td>
<td>b_3</td>
<td>~80 s</td>
</tr>
</tbody>
</table>
```

~10 s
~90 s
Parallel Model Reference Builds

![Diagram showing the parallel model reference builds with estimated build times (~80 s, ~90 s, ~10 s, ~30 s, ~10 s).]
Model Reference Parallel Build

User example

- Approximately 400 referenced models

Model Update Time comparison of first-time build with and without PCT

4 cores gives ~2.8 speedup

Does adding more cores yield more speedup?

- Original model: 3421.8
- With Parallel Build and 4 workers: 1212.8
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
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

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
## Summary

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Recommended Features</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit-Sim-Repeat</td>
<td>- Normal mode</td>
<td>Smart Editing</td>
</tr>
<tr>
<td></td>
<td>- Accelerator mode</td>
<td>Model Reference</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Simulation Data Inspector</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Solver Profiler</td>
</tr>
<tr>
<td>Tune-Sim-Repeat</td>
<td>- Performance Advisor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Fast Restart</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Accelerator + Fast Restart</td>
<td></td>
</tr>
<tr>
<td>Multi-Sim</td>
<td>- parsim</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- parsim + Accelerator + Fast Restart</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- parsim + Rapid Accelerator + Up-To-Date-Check-Off</td>
<td></td>
</tr>
</tbody>
</table>
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