MathWorks
**AUTOMOTIVE CONFERENCE 2024**
North America

# How Cloud-Based Virtual Vehicles help you Build Next-Gen Software

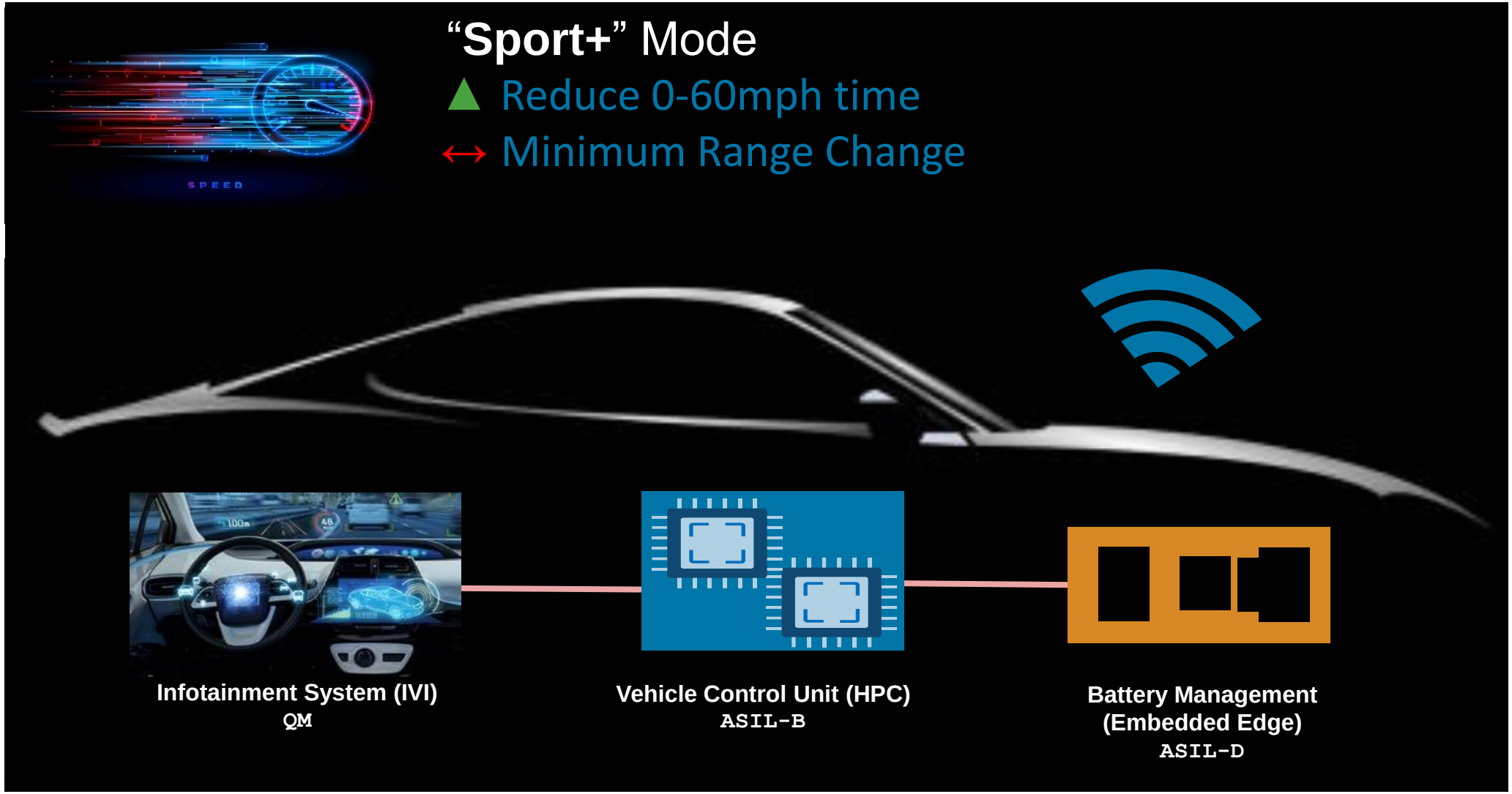
*Sameer K Muckatira, MathWorks*

(He/Him)

MathWorks

# Motivation

## Can we design new software without any vehicle/HiL time?



# Key Takeaways

**Goal:** *Empower engineers to assess the impact of software changes, unhindered by hardware access limitations.*

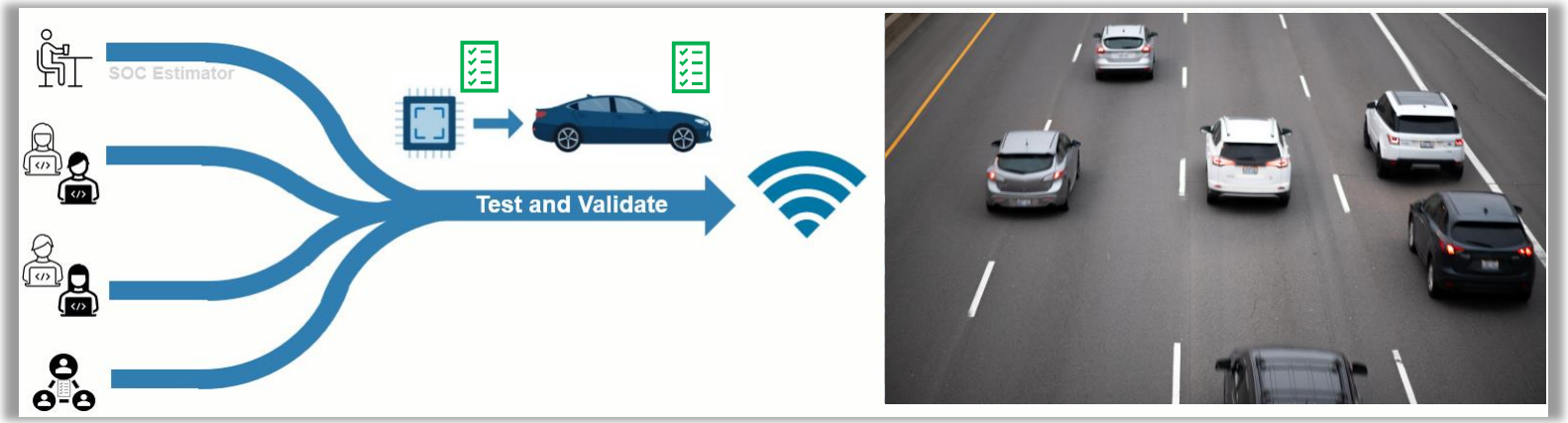
- SDVs lead to more frequent software releases, increasing vehicle complexity and systemic interactions.
- Virtual vehicle simulations allow you to assess software changes on system performance, minimizing test-cell and in-vehicle testing needs.
- Scale up simulation runs on the cloud to efficiently evaluate design alternatives faster.
- Vehicle and physics models can be re-used to test your production intent software stacks – in MiL, SiL, and virtual ECU testing.

# Agenda

**Goal:** *Empower engineers to assess the impact of software changes, unhindered by hardware access limitations.*

- Challenges of SDV Software Development
- Building Virtual Vehicle Simulations
- Scale Up Simulations – Moving from Desktop to the Cloud
- Reuse Virtual Vehicles Models for Production Testing

# What is a Software-Defined Vehicle?




Brand-distinctive features and main value for the customer **will come from Software**

Software development extends beyond production start, with **continuous updates** delivered throughout the vehicle's lifetime.

# Problem Statement

## What slows down your software planning and release?

- Increasing complexity of E/E architectures
- More software = more unexpected interactions
- Coordinate work between multiple global teams
- Scarcity of test cell time / vehicle testing time

**…. and now do this on an evergreen basis**

# Problem Statement

## What slows down your software planning and release at **a larger scale**?

- Increasing complexity of E/E architectures
- More software = more unexpected interactions
- Coordinate work between multiple global teams
- Scarcity of test cell time / vehicle testing time

**…. and now do this on an evergreen basis**

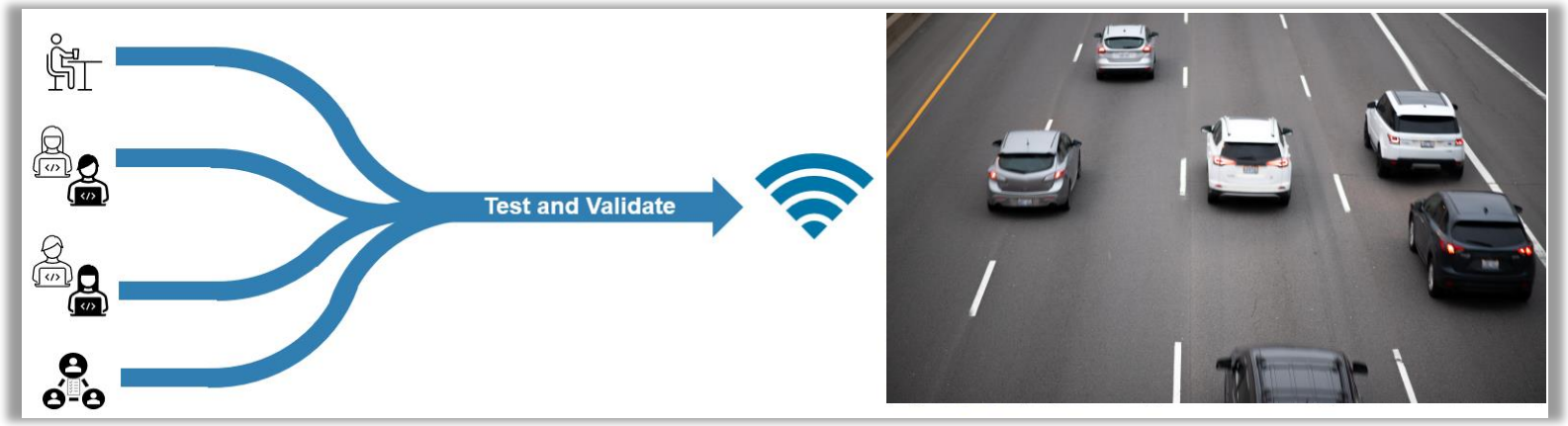
**A lot more interactions!**

**A lot more software!**

**A lot more people!**

*The continuous cycle of software releases and the expansion of global engineering teams elevate traditional vehicle development challenges into persistent, dynamic obstacles*

# Addressing the Challenges




**Develop faster without compromising on software quality?**

**Virtualize systems for analysis and testing**

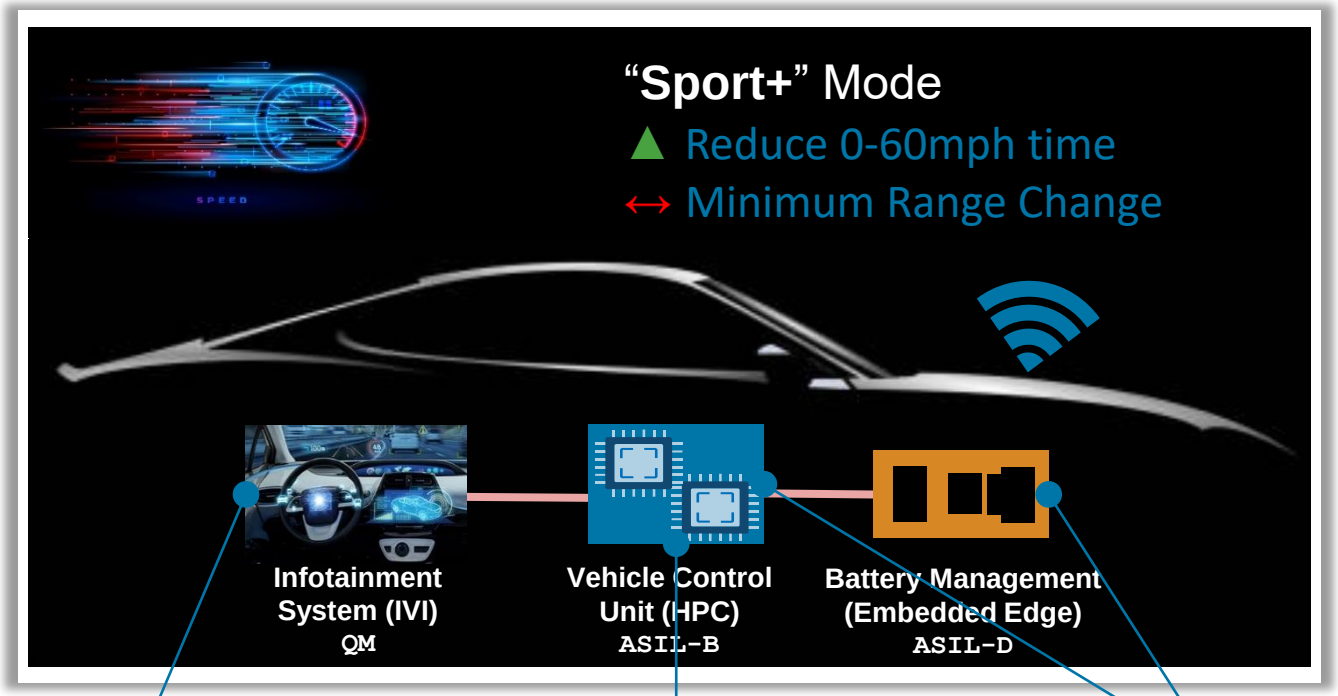
Build Virtual Vehicles to the right level of fidelity, integrating physics and software models

**Automate and scale on the cloud**

MATLAB and Simulink can be deployed to cloud environments for analysis and testing work

# EV Sport+ Mode

## Stakeholder needs - *How will it function?*




**Driver Requests for Sport+ Mode**

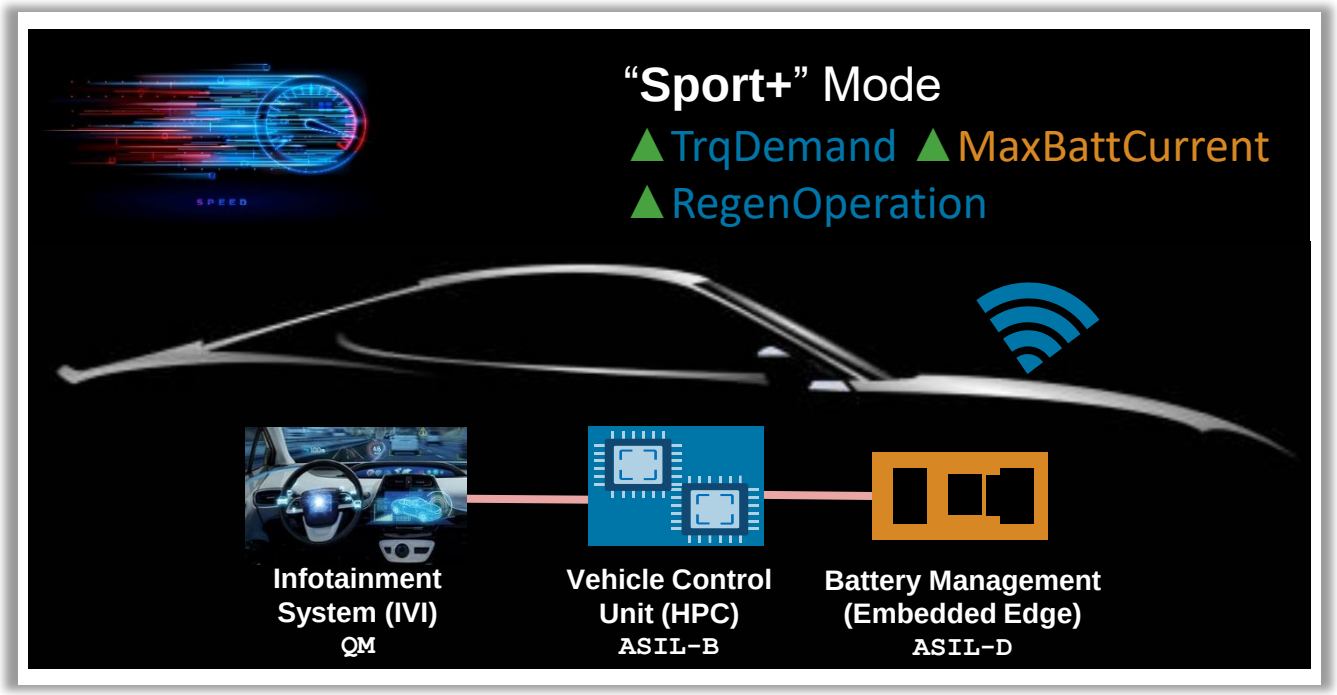
**Confirm no faults, Confirm Battery > 25%**

**Increase available power**

**How much?**
**What's the penalty on range?**
**Will my battery overheat?**

# EV Sport+ Mode

## Stakeholder needs → Engineering insights




**Challenge:**

| Performance goals |
|---|
| 0-60 mph in <6 seconds |
| Quantify loss in Range |
| Quantify gains from regen in different driving conditions |

| Variables |
|---|
| Maximum Battery Current Lim (A) |
| Brake-Regen Operation Limits (mph) |

*How much can we learn this without access to a vehicle?*

# Agenda

**Goal:** *Empower engineers to assess the impact of software changes, unhindered by hardware access limitations.*

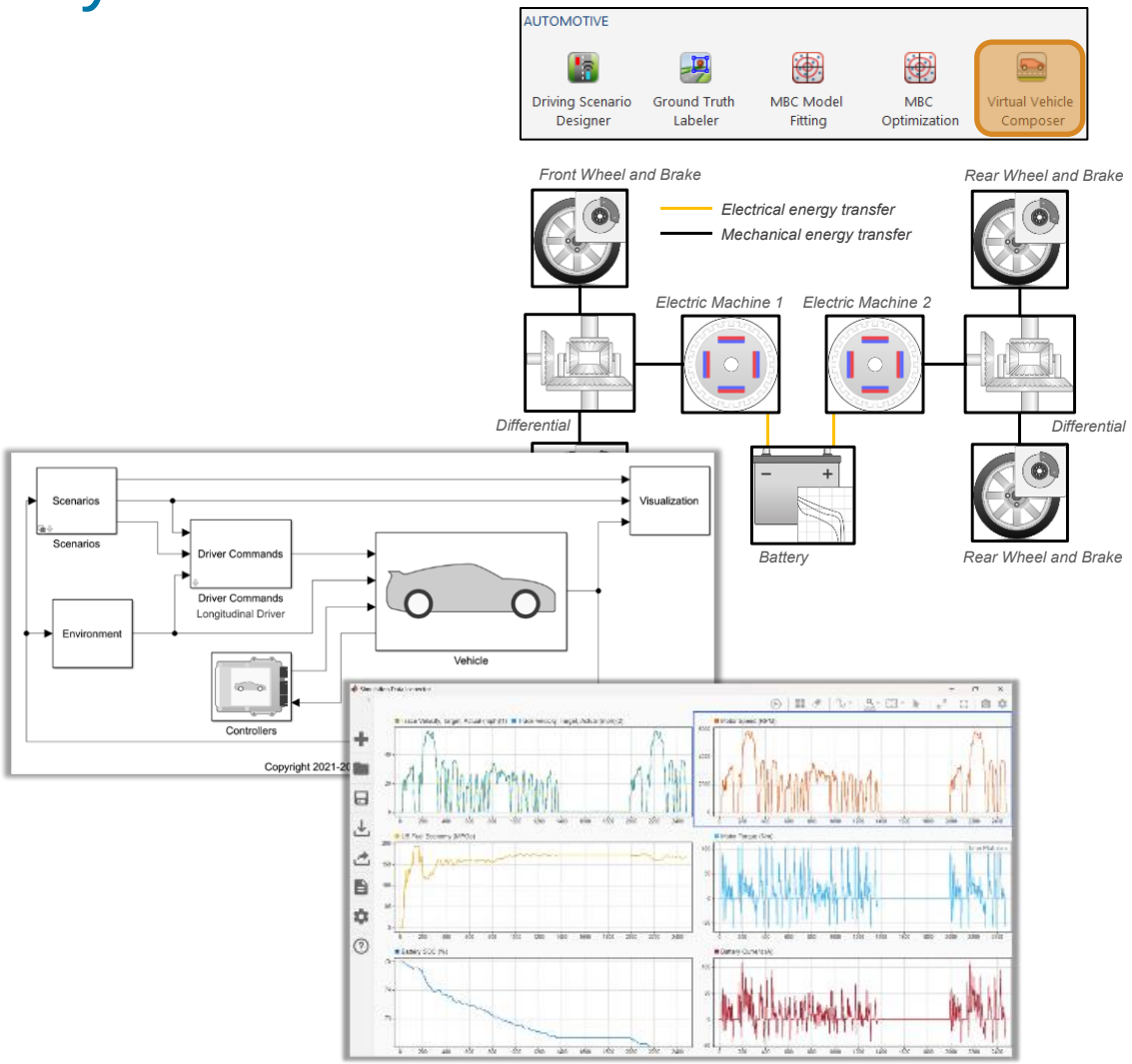
- Challenges of SDV Software Development
- Building Virtual Vehicle Simulations
- Scale Up Simulations – Moving from Desktop to the Cloud
- Reuse Virtual Vehicles Models for Production Testing

# Virtual Vehicle: Build Your Models Quickly

## Constructing a dual-motor EV model from nothing

### The Virtual Vehicle Composer App

- Quickly build and run a virtual vehicle
  - Based on libraries from Powertrain Blockset, Vehicle Dynamics Blockset & Simscape
  - Specify vehicle architecture, input datasheet details and calibration parameters
- Constructed model Includes calculations for energy consumption
- Access to prebuilt drive cycles

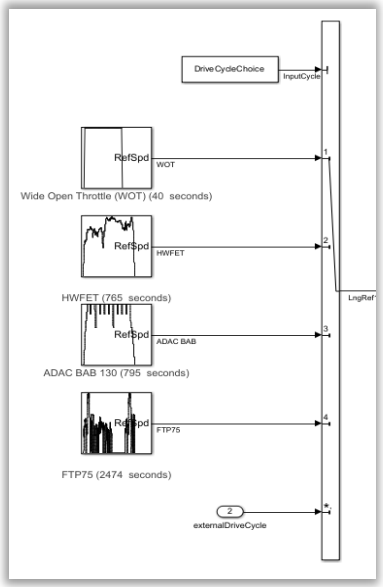



» Learn more: Virtual Vehicle Composer, DriveCycle Data

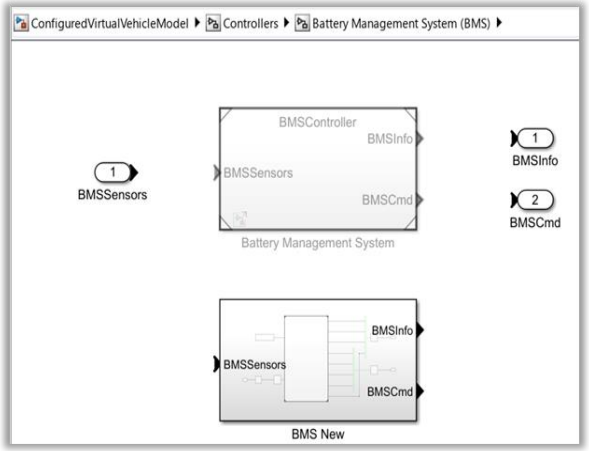
# Virtual Vehicle: Customize Your Virtual Vehicle

## Update your model based on the fidelity you require

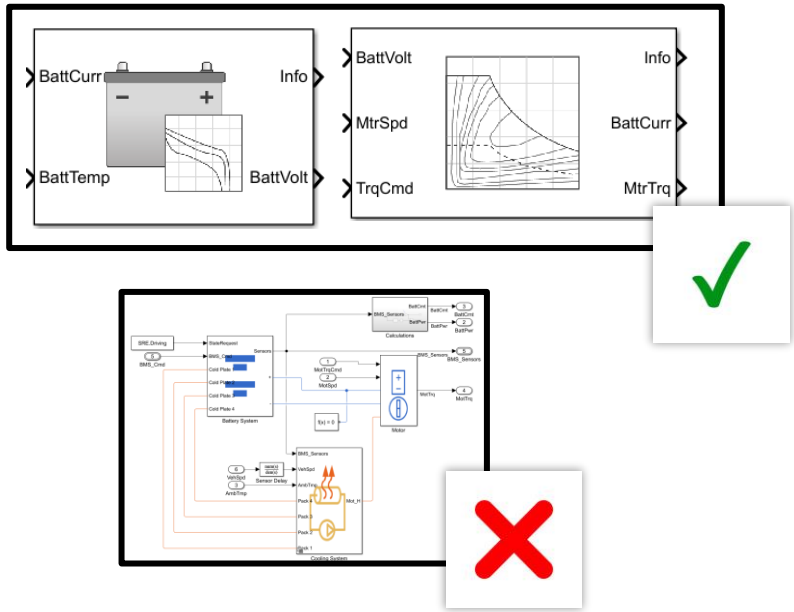
- Leverage Simulink's large scale modeling features to easily add details to your virtual vehicle




Choose between available drive cycles or input your own data




Custom BMS added as a variant subsystem

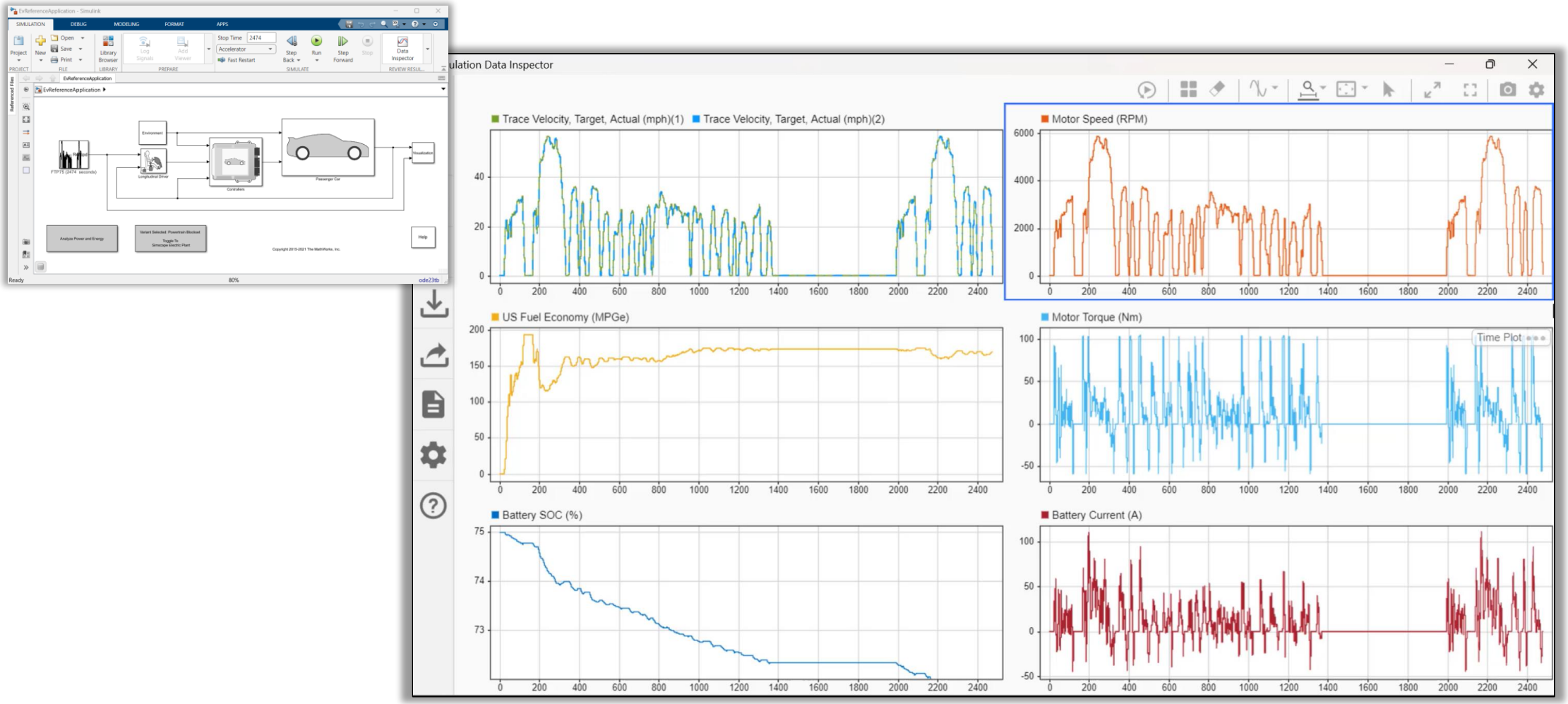



Speed over accuracy: Used Simulink powertrain blocks instead of Simscape

» Register here: Best Practices for Modeling Complex Systems (Upcoming Seminar Series)

# Virtual Vehicle: Running Simulations

## What can I learn from simulation?




» Learn more: Virtual Vehicle Composer, DriveCycle Data

# Agenda

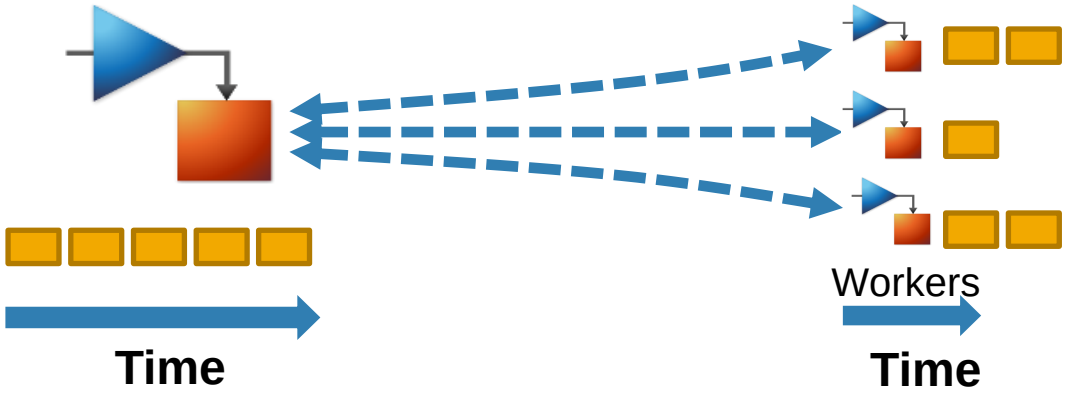
**Goal:** *Empower engineers to assess the impact of software changes, unhindered by hardware access limitations.*

- Challenges of SDV Software Development
- Building Virtual Vehicle Simulations
- Scale Up Simulations – Moving from Desktop to the Cloud
- Reuse Virtual Vehicles Models for Production Testing

# The Need to Scale Up

| | |
|---|---|
| Running 4 Simulations<br>(1 for each Drive Cycle)<br>  | 200 seconds |
| Running 1352 Simulations<br>(4 DriveCycles x 13 MaxDschrgCur Values x 26 RegenStrt Values)<br>`input_MaxDchrgCurrLimit = single(-12 : -0.5: -18)';`<br>`input_RegenStrt = (2.5:0.1:4)';`<br>`input_DriveCycleChoice = [1 2 3 4]';`<br>`input_DriveCycleChoiceTime = [40 765 795 2474]';` | ~18 hours |

- Parallelizing this work becomes essential as you scale up the number of Simulations



# Leveraging the `Simulink.SimulationInput` Object

- The Simulink.SimulationInput object is a useful way to define specific changes to be made for each simulation
- Defining the makes it easy to add conditions, and enables you using `parsim`

Specify Variable Values

Specify DriveCycle

Adjust Simulation Parameters

```
runID = 1;
clear in
for v1 = 1:length(input_MaxDchrgCurrLimit)
    for v2 = 1:length(input_RegenStrt)
        for v3 = 1:length(input_DriveCycleChoice)

            in(runID) = Simulink.SimulationInput(mdl);

            in(runID) = in(runID).setVariable('MaxSprtDchrgCurrLimit',input_MaxDchrgCurrLimit(v1),'Workspace','global-workspace');
            in(runID) = in(runID).setVariable('RegenSprtStrt',input_RegenStrt(v2),'Workspace','global-workspace');

            in(runID) = in(runID).setVariable('DriveCycleChoice',input_DriveCycleChoice(v3),'Workspace','global-workspace');

            in(runID) = in(runID).setModelParameter("StopTime",num2str(input_DriveCycleChoiceTime(v3)));

            runID = runID+1;
        end
    end
end
```

Workspace

| Name | Value |
|---|---|
| in | *1x1352 SimulationInput* |

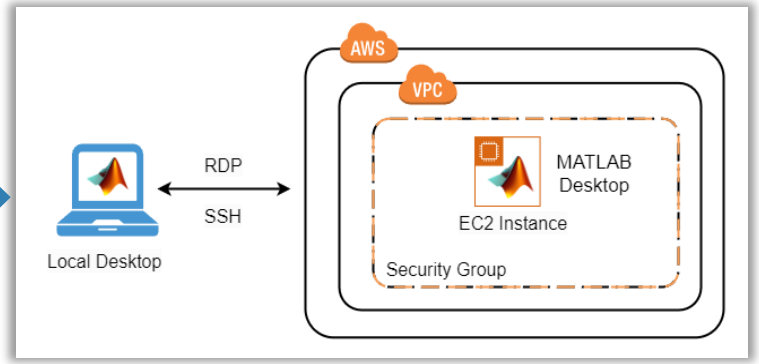
# Scaling up with `parsim` on the Cloud

## Different cloud computing resources for different jobs

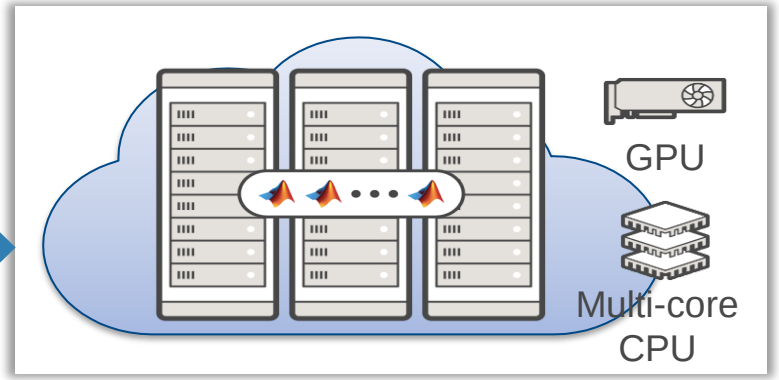
MathWorks Reference Architectures
Launch Stack
aws

```
simOut = parsim(in)
```

**Parallel Computing Toolbox**
**MATLAB**




**MATLAB on AWS EC2**




**MATLAB Parallel Server**

| Running 1352 Simulations |
|---|
| ~ **18 hours** in series |
| ~ **5.2 hours** on Quadcore Laptop |
| ~ **59 mins** on an m5.12xlarge EC2 instance, 24 core |

Worker Machine = m5.12xlarge (24 cores)

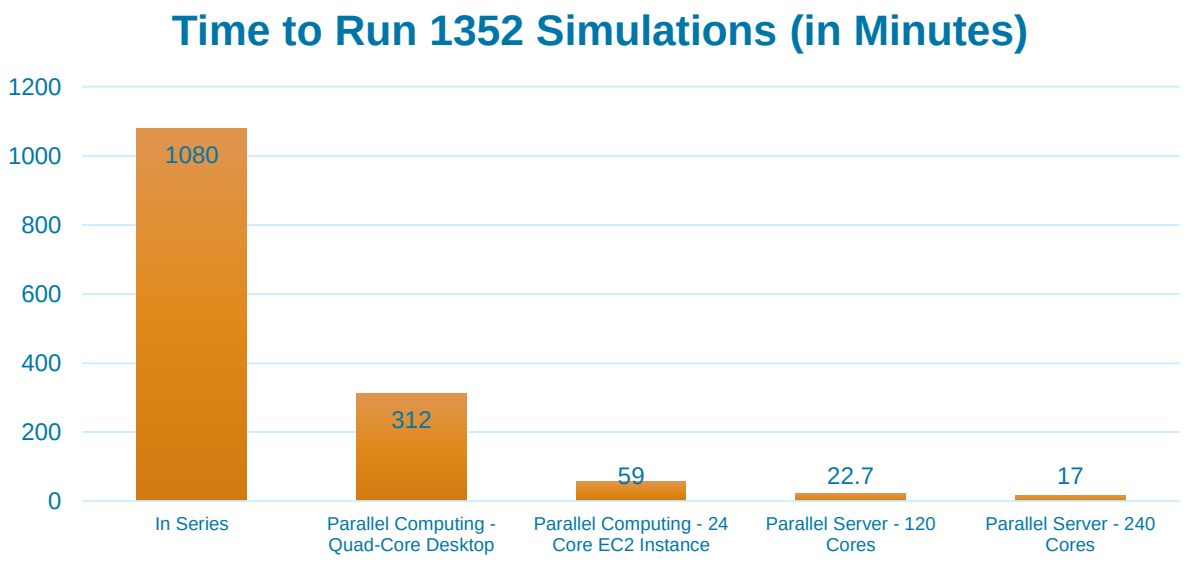
| Running 1352 Simulations |
|---|
| ~ **22.7 mins** on 5 Worker machines, 120 cores |
| **~17 mins** on 10 Worker machines, 240 cores |

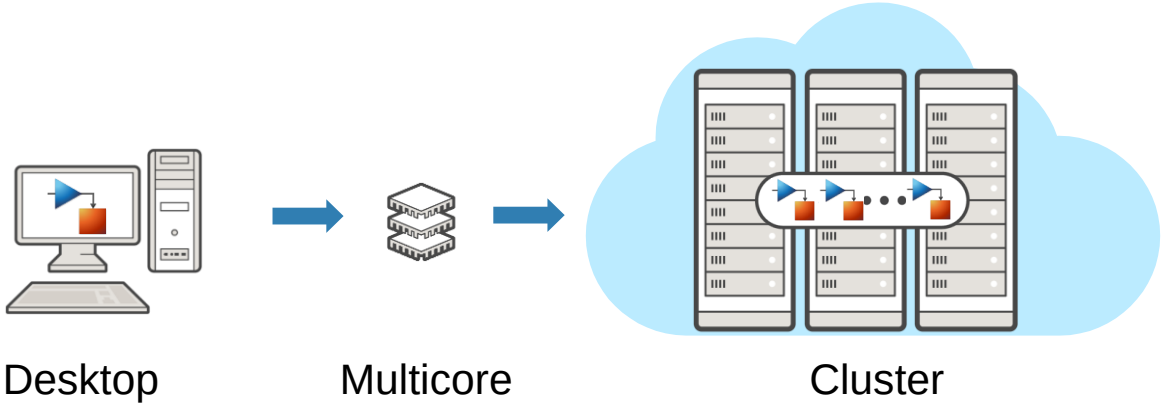
» Learn more: MATLAB on AWS

# Scaling up with `parsim` on the Cloud

## Takeaways

- Move from Desktop to the cloud with minimal code changes
- Get setup and easily use cloud resources with prebuilt reference architectures
- Choose the right computing resource based on your simulation requirements



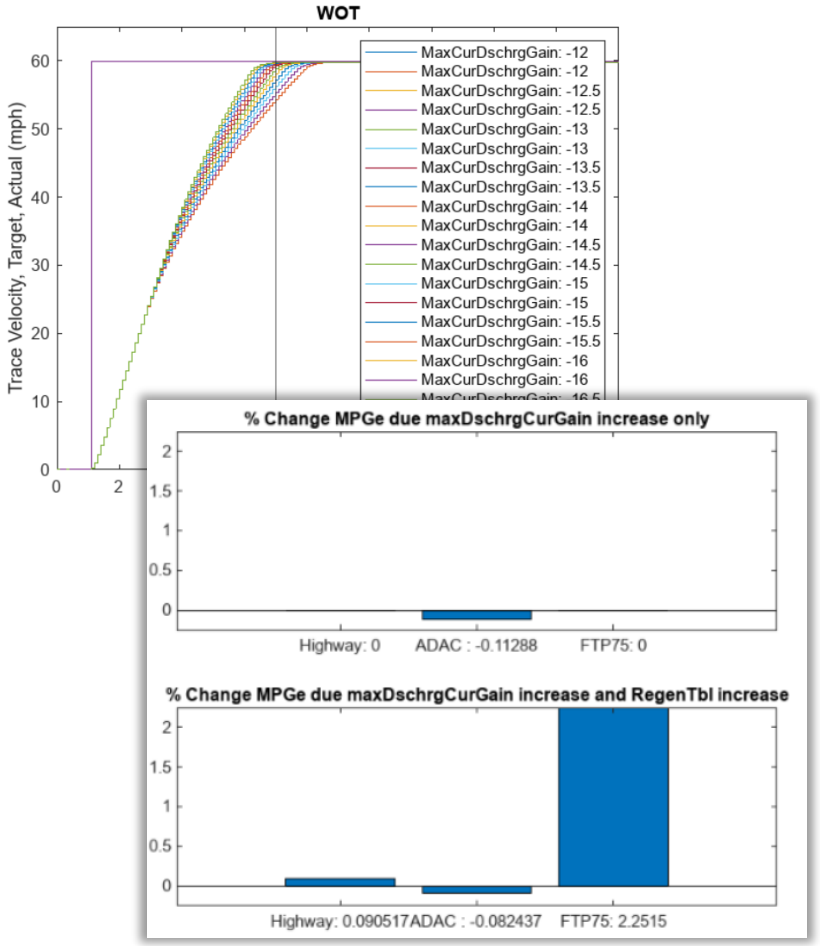

» Learn more: MathWorks Reference Architectures

# EV Sport+ Mode

## Lessons from the multi-simulation study

- Increasing the allowed **discharge current by from 10 to 17 (35%) can result in a 0-60 time of under 6 seconds.**
- The increased current results in higher energy consumption in high acceleration events, not highway driving.
- Highway cycles or cycles without a lot of braking do not see any gains from regenerative braking.
- Hard braking, as seen in WOT result a gain in MPGe

| Study Results: Recommendations | | | |
|---|---|---|---|
| Max Discharge Current Gain | ▲ | From<br>To | 10<br>17 |
| Regen Operation During Braking | ▲ | From<br>To | 5 - 9<br>2.5 - 9 |
| 0-60mph time | ▼ | From<br>To | 6.9 secs<br>6 secs |



# Agenda

**Goal:** *Empower engineers to assess the impact of software changes, unhindered by hardware access limitations.*

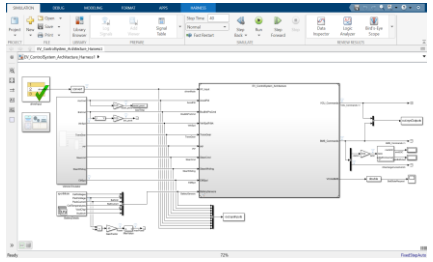
- Challenges of SDV Software Development
- Building Virtual Vehicle Simulations
- Scale Up Simulations – Moving from Desktop to the Cloud
- Reuse Virtual Vehicles Models for Production Testing

# From Analysis Models → Production Software Testing

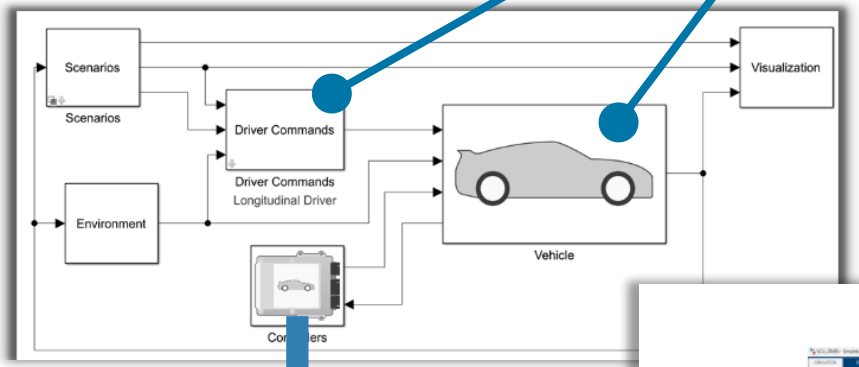
- Data from Virtual Vehicle Simulations can be used to create test cases

**Export physics data as CSV For Virtual ECU test inputs**

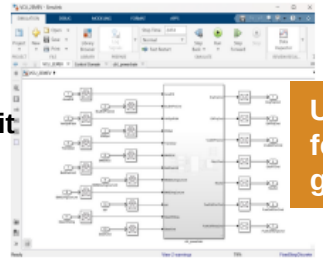
**Extract Data to act as Test Inputs (for MiL and SiL)**



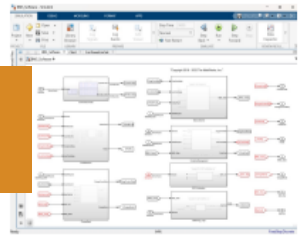
**BMS+VCU Test Harness**




**Vehicle Control Unit (VCU)**



**Update models for code-generation**

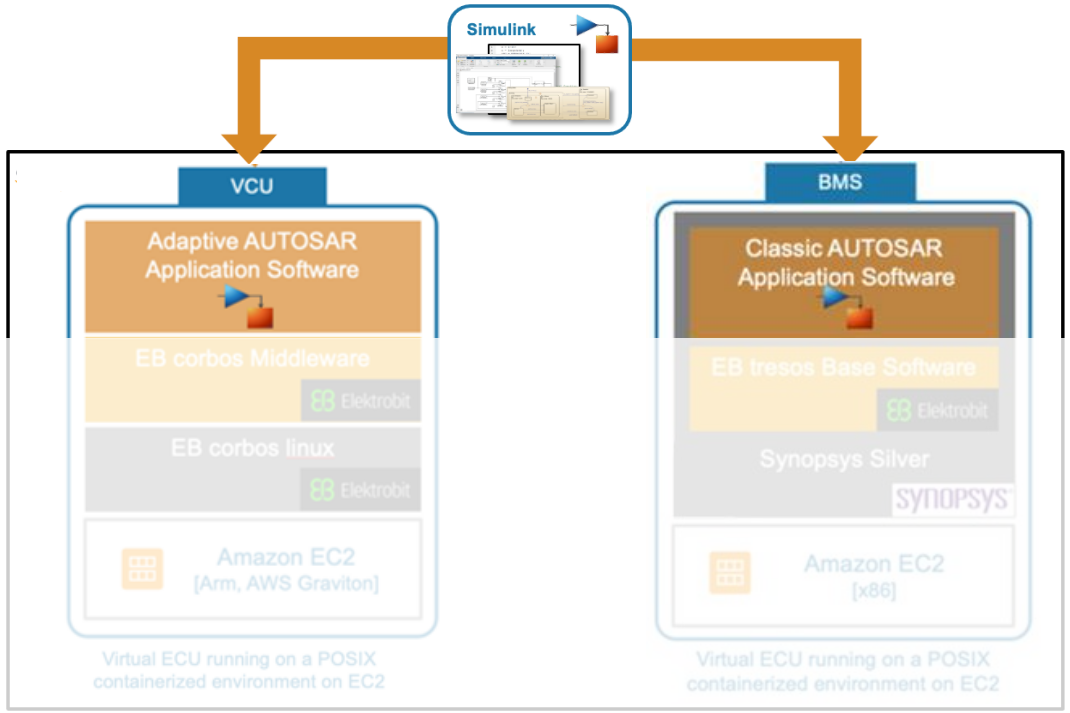


**Battery Management System (BMS)**

# From Analysis Models → Production Software Testing

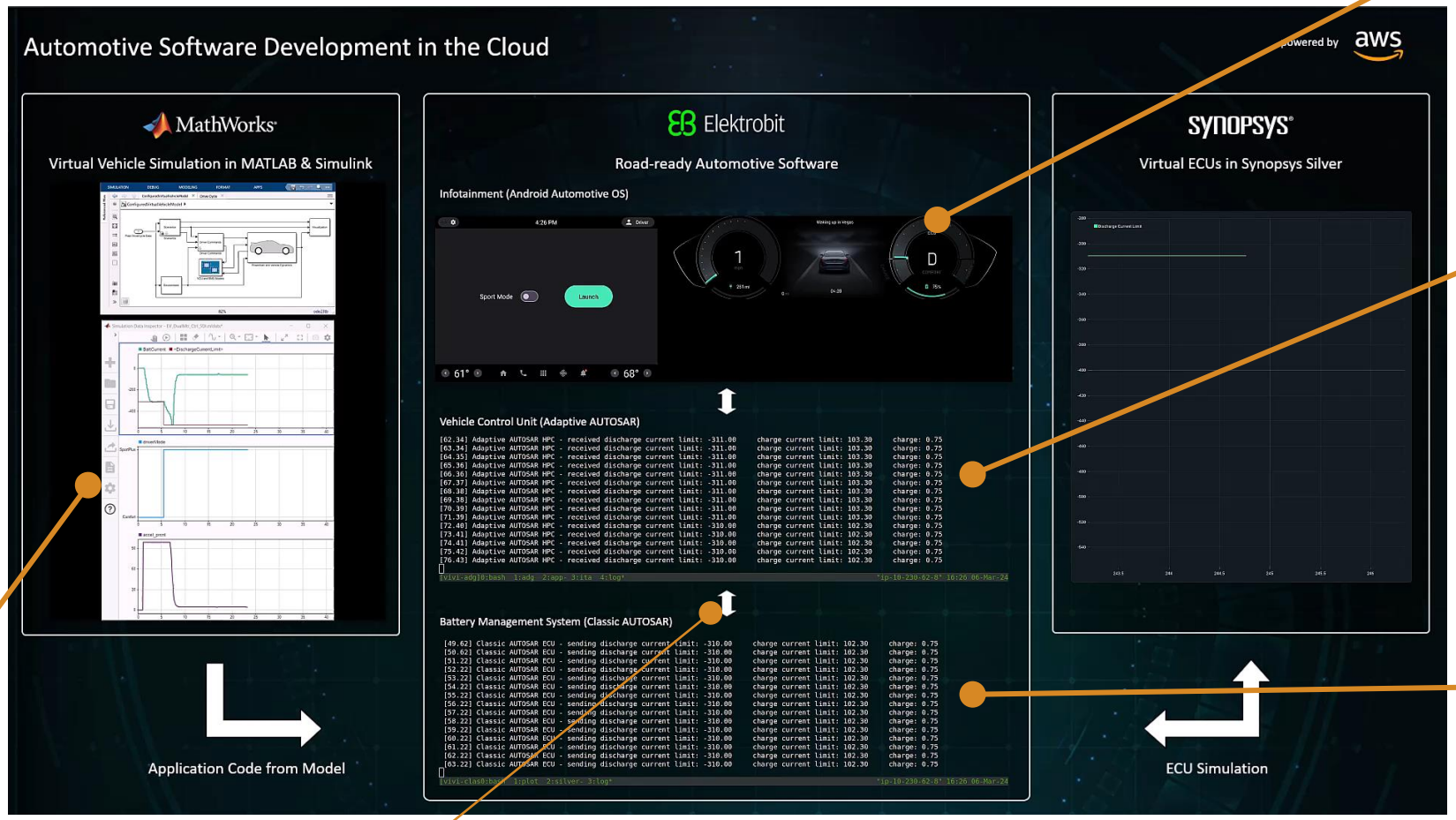
## Develop and deploy level 3 virtual ECUs to the cloud

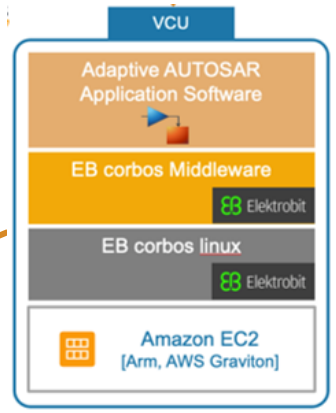
- Customize application code interface for specific middleware/hardware
  - Classic AUTOSAR
  - Adaptive AUTOSAR / DDS / ROS
- Verify the application software functionality using Simulink Test (MiL and SiL testing)
- Integrate application code with road-ready middleware to deploy within **Level 3** virtual ECUs
- Test virtual ECUs in a cloud native environment, using inputs from the Virtual Vehicle Simulation



# From Analysis Models → Production Software Testing

## Test level 3 virtual ECUs on the cloud





**Test Vectors & Vehicle Behavior exported from Simulink (Injected into vECUs via SOME/IP)**

**Inter ECU Communication (via SOME/IP)**

# Agenda

**Goal:** *Empower engineers to assess the impact of software changes, unhindered by hardware access limitations.*

- Challenges of SDV Software Development
- Building Virtual Vehicle Simulations
- Scale Up Simulations – Moving from Desktop to the Cloud
- Reuse Virtual Vehicles Models for Production Testing

# Key Takeaways

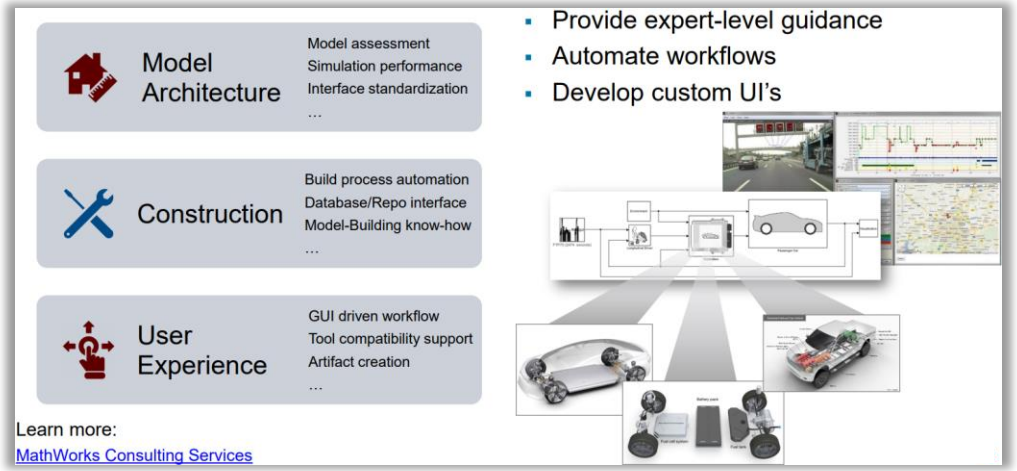
**Goal:** *Empower engineers to assess the impact of software changes, unhindered by hardware access limitations.*

- SDVs lead to more frequent software releases, increasing vehicle complexity and systemic interactions.
- Virtual vehicle simulations allow you to assess software changes on system performance, minimizing test-cell and in-vehicle testing needs.
- Scale up simulation runs on the cloud to efficiently evaluate design alternatives faster.
- Vehicle and physics models can be re-used to test your production intent software stacks – in MiL, SiL, and virtual ECU testing.

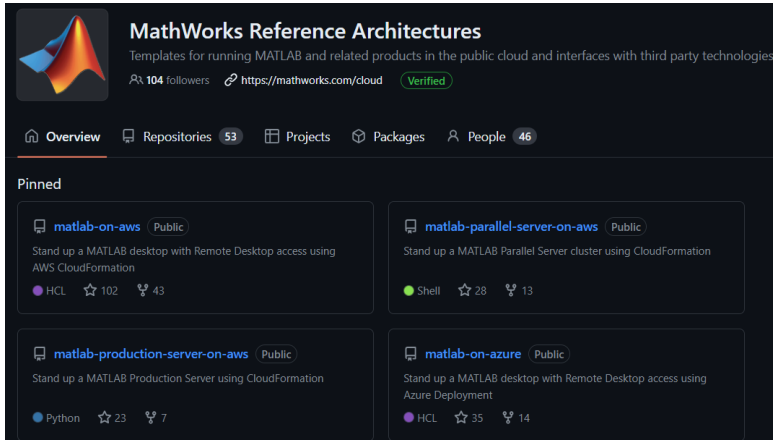
# MathWorks Resources

## Where can I find more information ?

[MathWorks Consulting for Virtual Vehicle Development]




[MathWorks Reference Architectures]




[Solutions Page: Software-Defined Vehicle]


**Ready to start your SDV journey?**

Scan the QR code to learn more!

MathWorks
**AUTOMOTIVE CONFERENCE 2024**
North America

## Reach out to learn further:

MathWorks®