EGADSlite: A Lightweight Geometry Kernel for Distributed Mesh Generation and Adaptation

Bob Haimes
haimes@mit.edu
Aerospace Computational Design Laboratory
Massachusetts Institute of Technology

Overview

- Introduction – Geometry & HPC
- Current bottlenecks
- Approach
  - The Engineering Sketch Pad: `ESP`
  - The Geometry Kernel (`EGADS`)
  - Construction of `EGADSlite`
  - Memory Footprint
  - Inverse Evaluation Timings
  - Threading
  - `EGADSlite` Assessment
- Parallel/Distributed Examples
  - Distribution of Software Components
  - `REFINE`'s Use of `EGADSlite`
- Status

Introduction – Geometry & HPC

CFD 2030

- Section 5.1 (Effective Utilization of HPC)
  "Lack of scalable CFD pre- and post-processing methods"
- Section 5.3 (Autonomous and Reliable CFD Simulations)
  "Mesh generation and adaptivity"
  - "inadequate linkage to CAD"
  - "poor mesh generation performance and robustness"

Scalability of the Entire Process

Amdahl's Law tells us that any serial portion of the *application* will be the limiting factor in scalability.

Current bottlenecks

Generally:

- The grid generation systems frequently operate on one HUGE front-end machine; the grid is then partitioned and distributed throughout the HPC system.
- The underlying geometry system is not distributed
  - Geometry systems were **not** built to exploit parallel and/or distributed computing.
  - Most geometry systems are licensed, making their distribution across an HPC system impractical (if not prohibitively expensive).
- *In situ* adaptation requires the grid adaptation process to operate in a distributed manner. Moving vertices on or near the surface requires access to the geometry.
- Higher-order methods may require local surface slopes and curvatures in order to generate meshes with curved elements.

Engineering Sketch Pad (ESP)

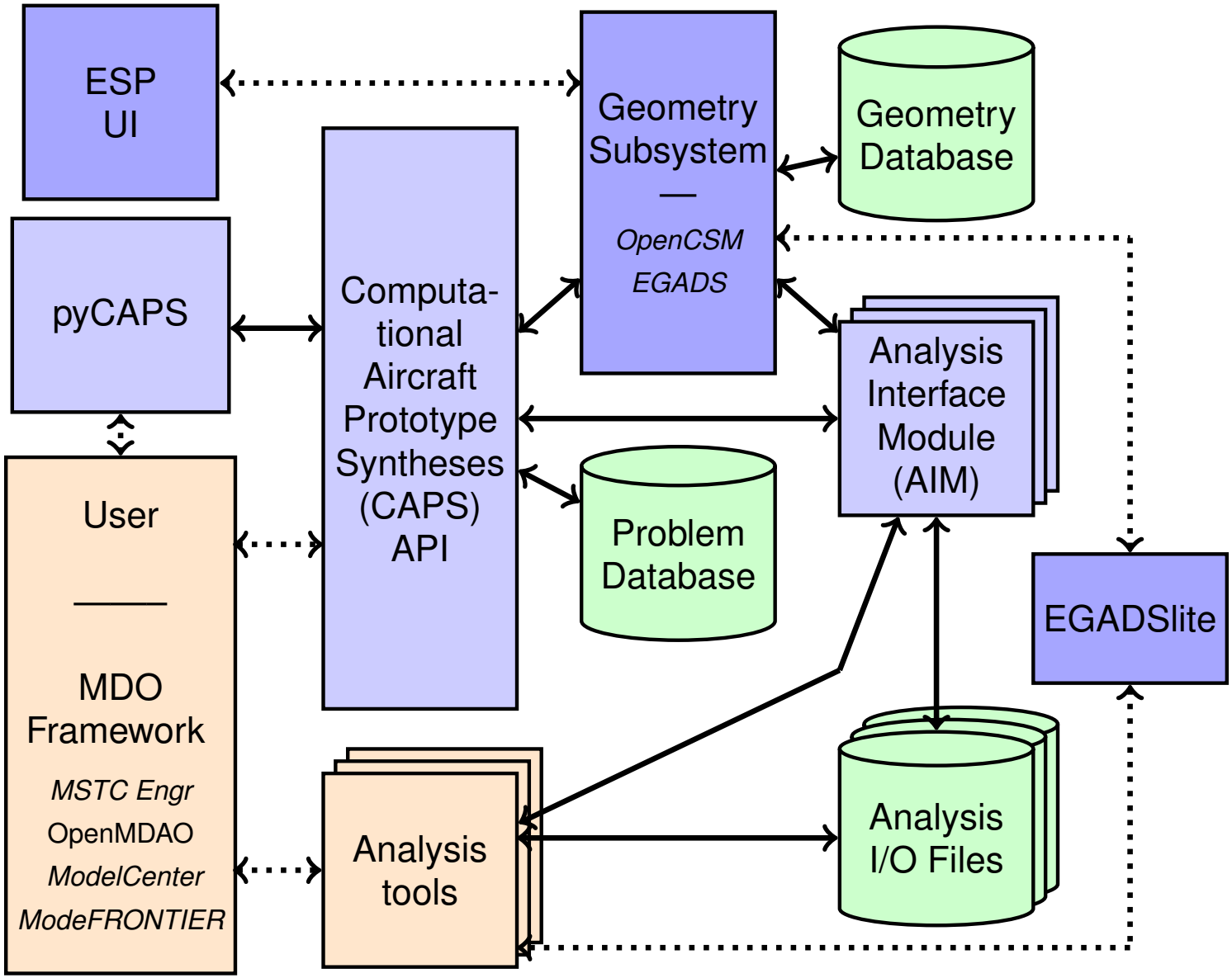
ESP is:

- a parametric geometry creation and manipulation system designed to FULLY SUPPORT multi-fidelity and multidisciplinary **a**nalysis (an **aCAD** system)
- can be embedded into other software systems to support their geometric and process needs (just a collection of APIs)
- working towards an integrated conceptual/preliminary design system of aerospace vehicles

ESP is not:

- a full-featured **m**anufacturing-based computer-aided design (**mCAD**) system
- software used for creating "drawings"
- an MDAO Framework

Engineering Sketch Pad (ESP) with CAPS



The Geometry Kernel (EGADS)

The Engineering Geometry Aircraft Design System (EGADS) is an open-source geometry API (depends on OpenCASCADE)

- Reduces OpenCASCADE's 17,000 methods to about 100 calls
  - Supports C, C++ & FORTRAN
- Full suite of geometric primitives
  - curve: line, circle, ellipse, parabola, hyperbola, offset, bezier, BSpline (including NURBS)
  - surface: plane, spherical, conical, cylindrical, toroidal, revolution, extrusion, offset, bezier, BSpline (including NURBS)
- IGES and STEP reader and writer
- Provides *bottom-up* and/or *top-down* construction
- Solid creation and Boolean operations (*top-down*)
- Provides persistent user-defined attributes on topological entities
- Adjustable tessellator (*vs* a surface mesher) with support for finite-differencing in the calculation of parametric sensitivities

Construction of EGADSlite

- Provide the portions of EGADS that are required for meshing
  - BRep parsing
  - Geometric evaluations and inverse evaluations
  - Containment predicates
  - Attribution
  - Tessellation – a good test
- No geometry building (read-only), but requires an EGADS app (such as ESP) to generate the internal data (byte *stream*)
- ANSI C with the ability to support C++ and FORTRAN easier porting to novel architectures (GPU)
- Same function signatures as EGADS with same side-effects except for:
  - EG_loadModel
  - EG_deleteObject

Memory Footprint

Determining runtime size is difficult and depends on the problem

- Use dynamic library size on disk as a surrogate
- Note: not all `EGADS`/OpenCASCADE libraries may be needed

	# of Libraries	size (Kbyte)
`EGADS`	52	65384
`EGADSlite`	1	342

Apple MACbook Pro running OSX 10.12.5 with Xcode 8.3.3 and OpenCASCADE version 6.6.0.

Inverse Evaluation Timings

Given XYZ – find closest point on a geometric entity solved by minimizing distance using Newton-Raphson

	EGADS 1.11		EGADSlite	
Case	nfail	time	nfail	time
demo2	0	0.40	0	0.04
tutorial1_whole	0	8.48	0	1.04
design2	0	0.35	0	0.04
design3	0	2.56	0	0.16
tutorial2	0	1.04	0	0.16
tutorial3	0	98.96	0	2.51
myPlane	20	765.86	0	24.24
bottle2	0	8.14	0	0.83
wingMultiModel	0	38.30	0	0.64
bullet	102	1.75	0	0.11
connect5	0	0.47	0	0.03
group2	0	1.55	0	0.14
hl-crm-gapped-flaps	0	17115.71	38	1804.50
hl-crm-sealed-flaps			16	1783.64
jsm_case01			0	150.30
jsm_case02			0	153.27

A “failure” indicates that the inverse evaluation completed, but that it found a different point on the surface than expected

Threading

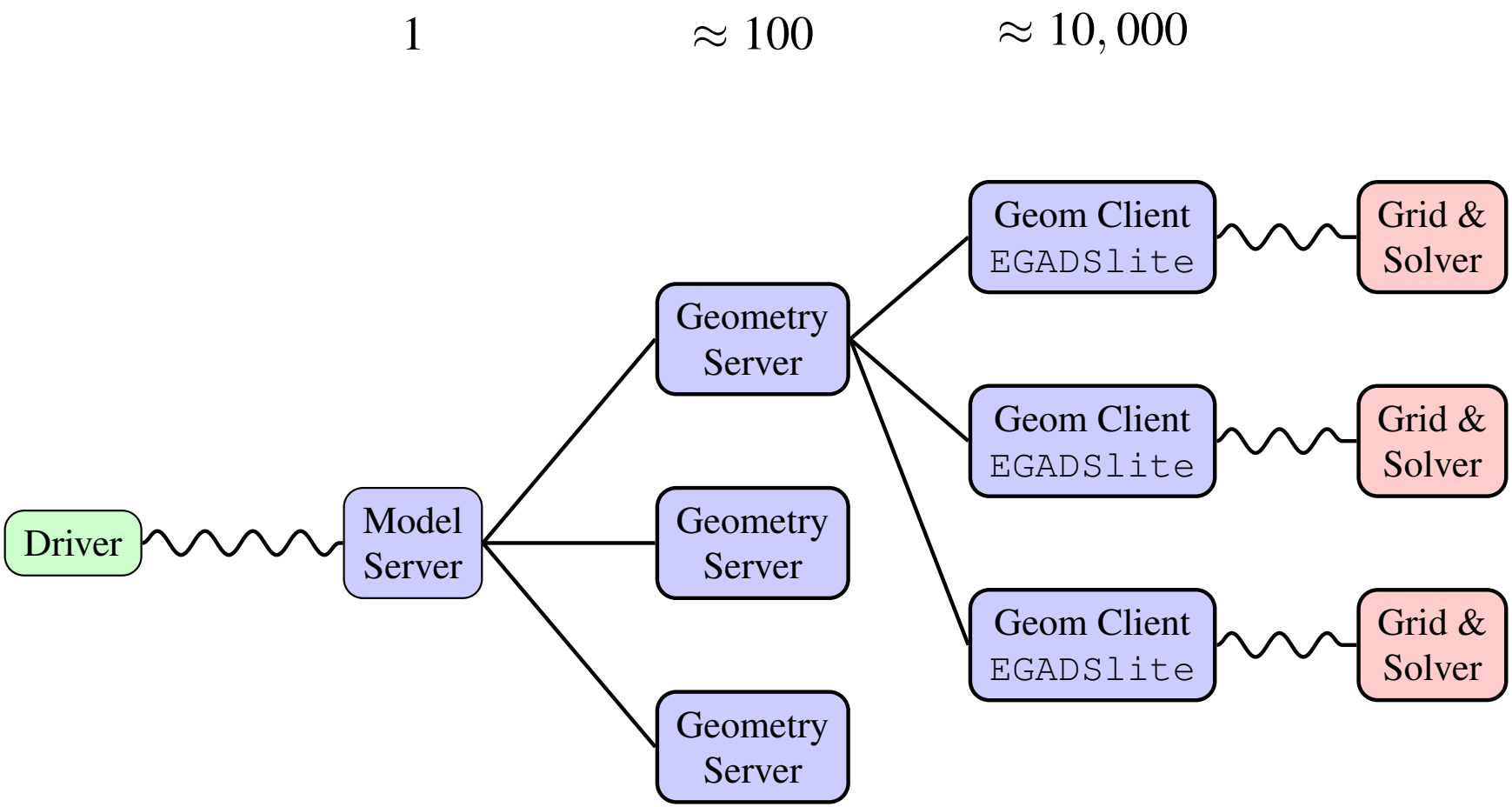
The timing of EGADS *hl-crm-gapped-flaps* case begs for threading!

- The *EGADS* tessellator avoids inverse evaluations (has always been threaded)
- In general, inverse evaluations should be avoided, but
- GMGW1 analysis (AIAA2018-0131) required inverse evaluations to check position of surface vertices – due to the lack of geometric information in the mesh files
- Threading these operations (per Face) using EGADS only 150% of the CPU utilization was realized and most was *system* time!
  - Conjecture: a critical portion of the OpenCASCADE inverse evaluation is bounded by a MUTEX giving something that is thread-safe but **not** scalable
- EGADSlite displays perfect scalability when threaded!

EGADSlite Assessment

- EGADSlite technologies have been (re)implemented in EGADS
  - Now more robust at some timing cost (better seeding – BSplines)
  - EGADS now performs the same as EGADSlite
  - Thread scalability!
  - Note: OpenCASCADE has improved it's scalability since Rev 7.0 but this performance issue has not been revisited
- EGADSlite has been part of ESP from Rev 1.12 on
  - No explicit message passing code
    - EGADS can serialize Model Objects into a *stream*
    - EGADSlite deserializes the *stream* to reconstruct the BRep Model Object hierarchy
  - Distribution has MPI examples:
    See $ESP_ROOT/externApps/Pagoda
  - GPU *Cuda* port in progress

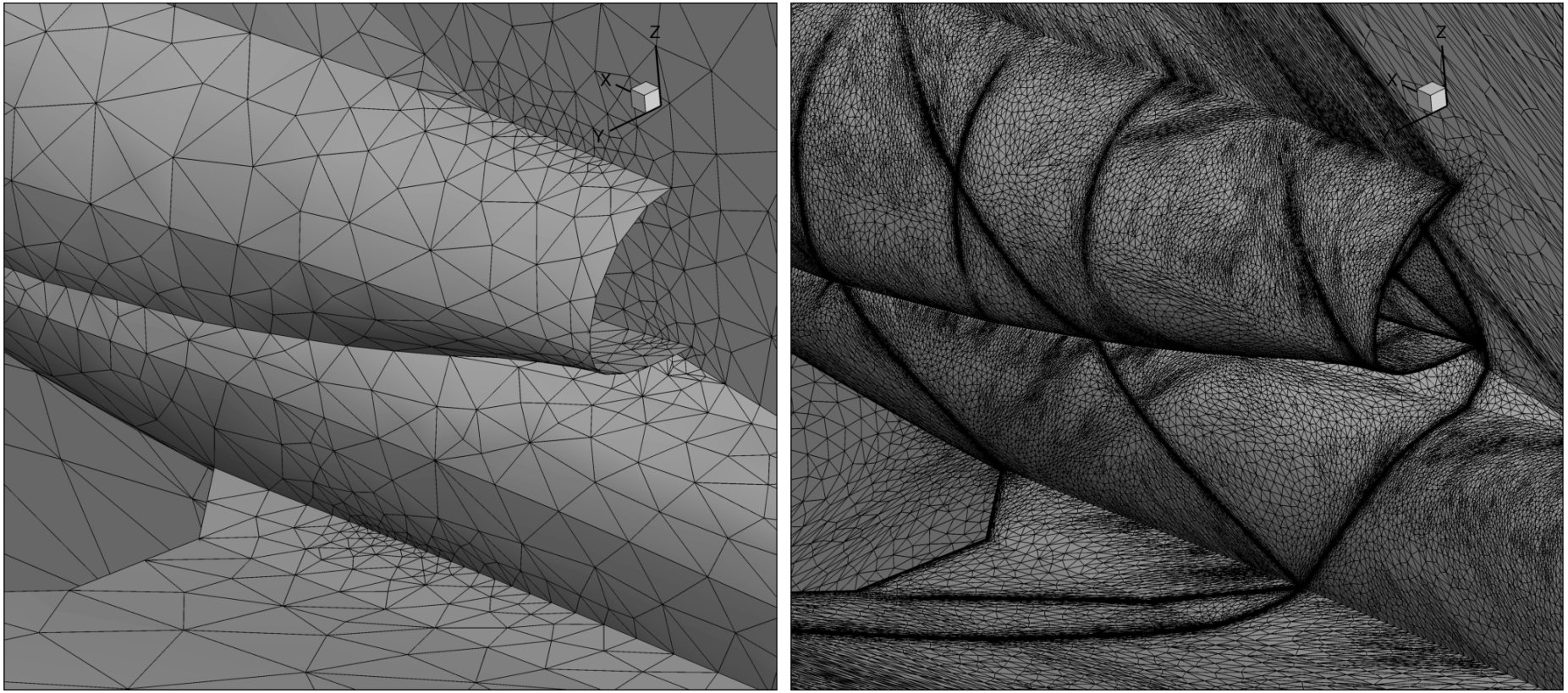
Distribution of Software Components



A possible distributed design system layout

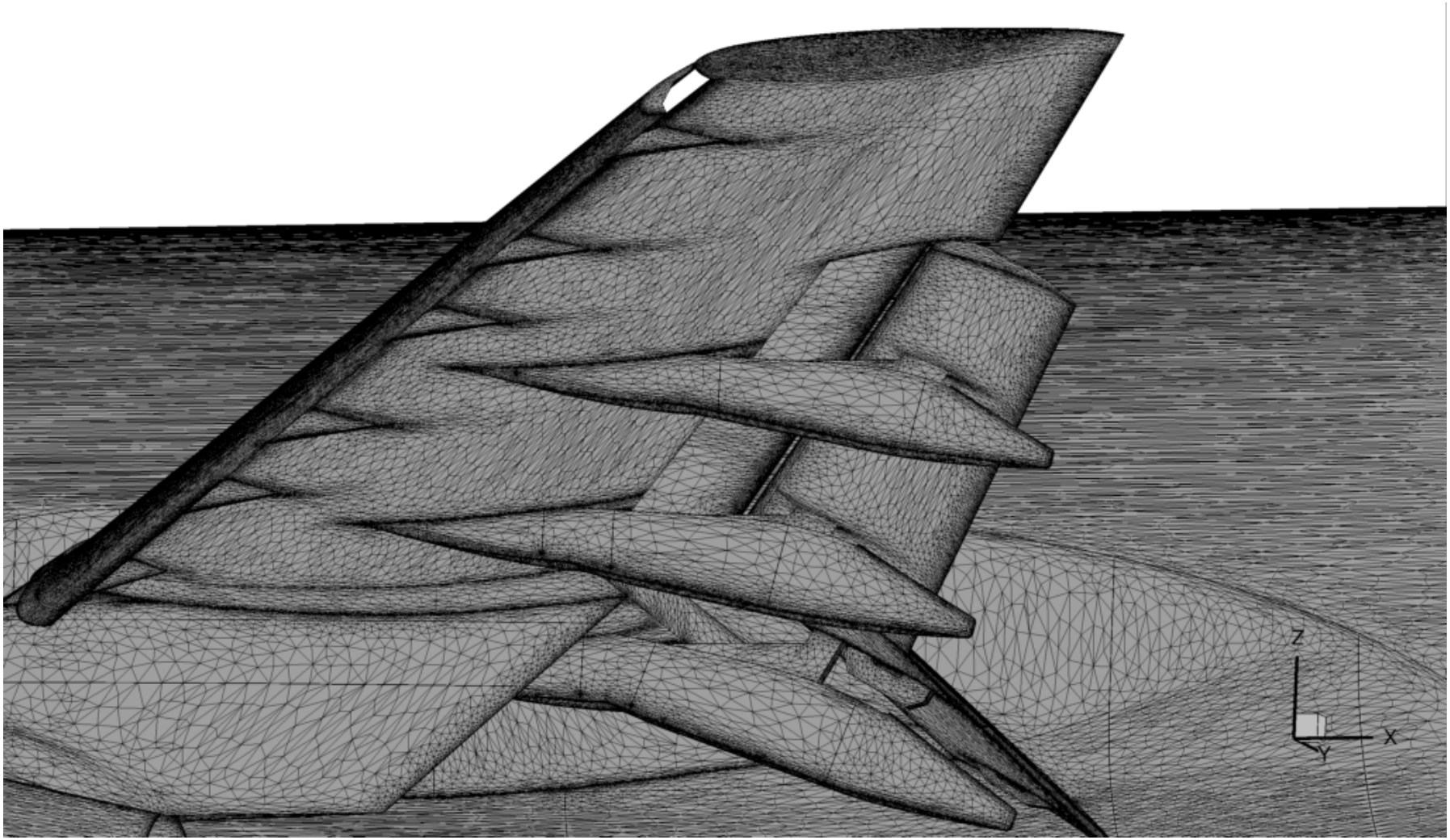
Note: no claims to be exascale!

REFINE's Use of EGADSlite



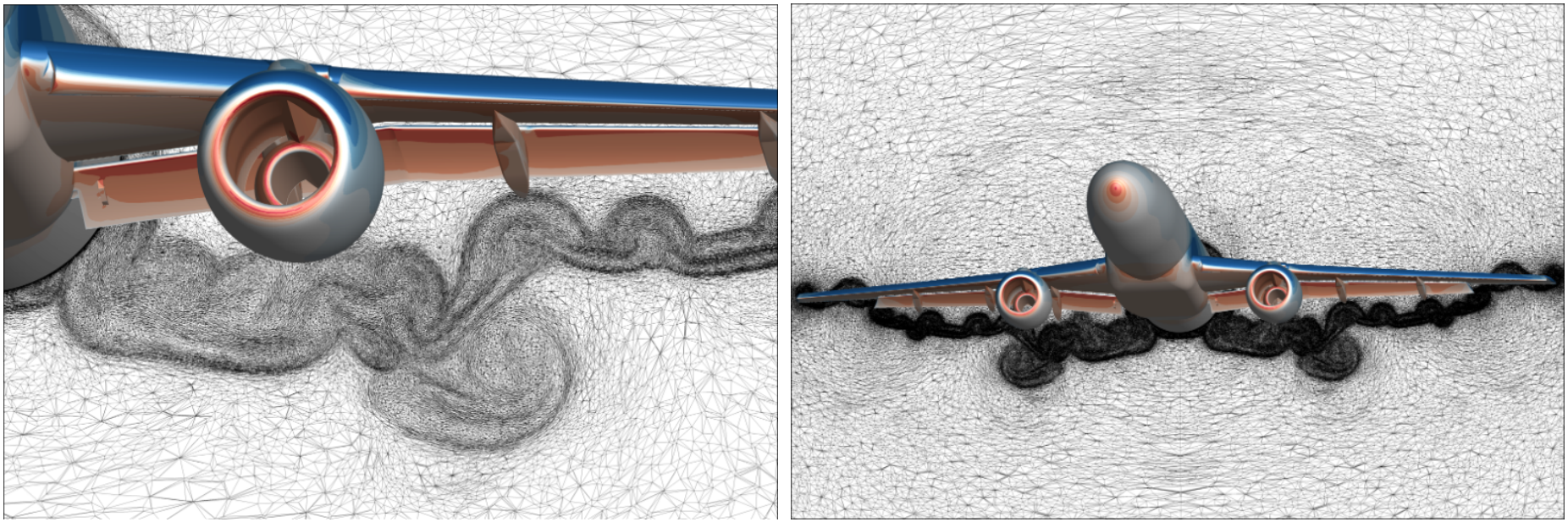
Initial EGADS tessellation of the C25F inlet region and REFINE's adapted mesh

REFINE's Use of EGADSlite



A blowup of a coarse adapted mesh of the JSM configuration

REFINE's Use of EGADSlite



JAXA's high lift configuration using NASA's Fun3D and REFINE

Status

Acknowledgements

- John Dannenhoffer (Syracuse U) & the rest of the `ESP` team
- Bill Jones (NASA LaRC) for the *Cuda* port of `EGADSlite`
- Mike Park (NASA LaRC) for `REFINE` & its results

`ESP`'s Direct Support

- PAGODA: **PA**rallel **GeO**metry for **D**esign and **A**nalysis
  Bill Jones (NASA LaRC), Technical Monitor
- EnCAPS: **En**hanced **C**omputational **A**ircraft **P**rototype **S**ynthesis
  Ryan Durscher (AFRL/RQVC), Technical Monitor

Software freely available at:

http://acdl.mit.edu/ESP