Can Legacy Codes Scale on Tens of Thousands of PEs or Do We Need to Reinvent the Wheel?

Robert F Tomaro, William Z. Strang and Kenneth E. Wurtzler

Cobalt Solutions, LLC

Over the last several years, there have been many discussions involving the potential scalability of legacy codes/numerics concerning their ability to fully utilize large core count machines. The prevailing thought and conclusion of those involved in the discussions were that legacy codes could not scale well on thousands of processors, let alone tens of thousands of processors. Therefore, new codes using new numerics had to be developed. In fact, a large DoD effort was established 5 or 6 years ago tasked with developing the new numerics required to fully utilize hundreds of thousands of processor machines. In light of the recently published scalability results of the DoD's effort at CFD code development, we decided to retest the scalability of *Cobalt* to see if the above conclusion and results held true for that legacy code.

I. Introduction

In the early 2000's, *Cobalt*[1,2] was used for benchmarking new HPC machines. This included scalability tests on *KRAKEN, jvn, eagle* and *sapphire*. NAVO received the 2048 processor IBM, *KRAKEN*, as a TI-04 machine[3]. The first Capability Application Projects (CAP) were completed on *KRAKEN* including an F/A-18E free-to-roll simulation using *Cobalt*. ARL received the 2048 processor linux-based machine, *jvn*, in TI-04 while AFRL received the 2048 PE SGI, *eagle*[4], as a TI-05 machine. Finally, *sapphire* was installed at ERDC from TI-05[5]. This machine was a Cray XT3 containing 4176 processors. At that time, *sapphire* was the most powerful supercomputer in DoD. The last time the scalability of *Cobalt* was performed for HPC was on *sapphire* in 2005.

Figure 1 shows the 2005 scalability results for *Cobalt* for *KRAKEN, jvn, eagle* and *sapphire*. The chart shows the super-linear speed-up of *Cobalt* on those four machines. Super-linear speed-up is defined as a scalability efficiency greater than 100%. Of particular interest are the results on *sapphire*. These results are often quoted as the best CFD scalability obtained by a production-level code. However, it is often said that *Cobalt* scales up to 5000 processors. There are two problems with this statement. First, the results on *sapphire* were on 4000 processors. Second, and more importantly, the phrase "scales up to" implies that *Cobalt* did not scale beyond 4000 processors. In actuality, we simply ran out of machine. The results on *sapphire* clearly show that scalability efficiency was still increasing at the processor limit. Additionally, the grid size is never specified when these results are mentioned. We will show later that the number of cells per processor is a critical parameter when measuring scalability efficiency. We state here that the grid used in the 2005 scalability study contained 6.5 million cells.

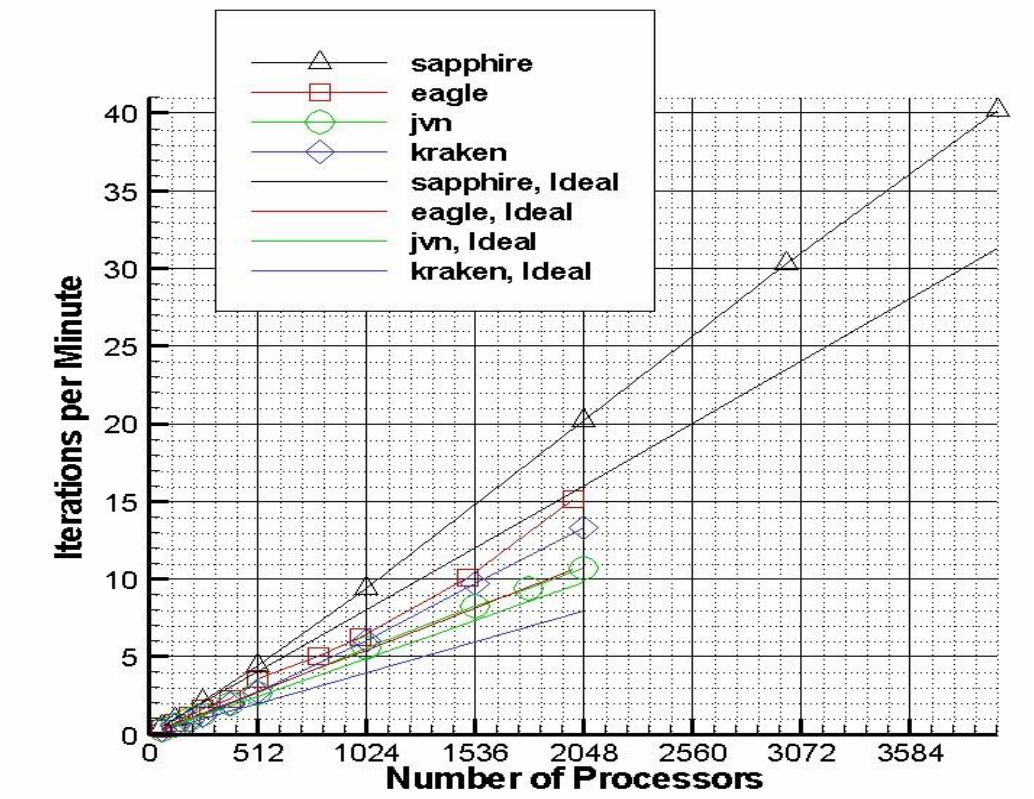

Figure 1: Scalability Results for *Cobalt* circa 2005

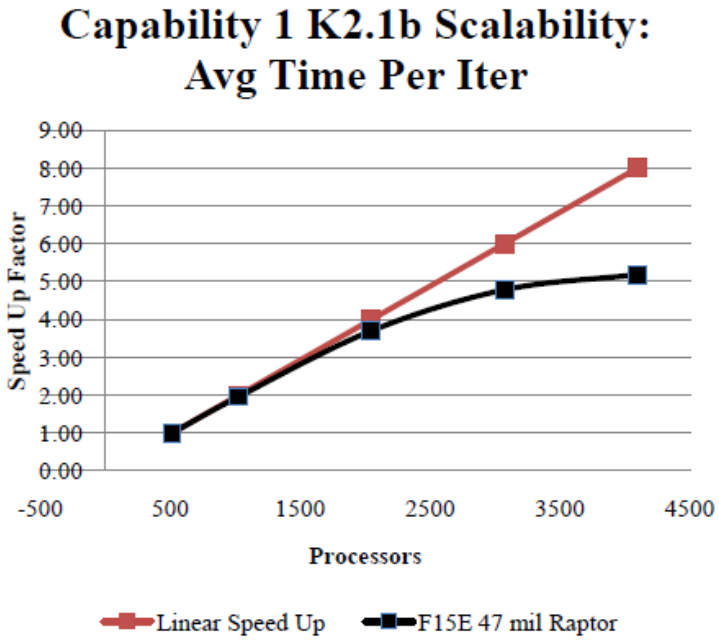

Figure 2: Scalability of *Kestrel* v2.1b in 2011

Figure 2 shows the scalability of CREATE's *Kestrel* v2.1b as presented by Morton[6] at the 2011 HPC User Group Conference. These results are for a 47 million cell grid on the Cray XE6, *raptor*, located at AFRL. The CFD solver in *Kestrel* v2.1b is kAVUS, which is based on the legacy code AVUS from AFRL[7]. The

efficiency of *Kestrel*'s scalability is under 100% on 2048 PEs. On 4096 PEs, *Kestrel*'s scalability is 65% which was deemed acceptable by Morton, et al.

II. Current Scalability Results for *Cobalt*

We decided to update the *Cobalt* scalability charts for a few reasons. First, it has been seven years since the benchmarking was performed on *sapphire*. Second, there have been many modifications to the internal workings of *Cobalt* in the time since. Third, machine architecture has changed from single-core machines to multi-core machines, which could easily affect scalability results. Lastly, the current scalability results of *Kestrel* v2.1b on *raptor* show, in our opinion, poor scalability. This could imply that *Cobalt*, another so-called legacy code, will not scale well on the current architectures at HPC, which must be investigated. We will present scalability results on three different single-grid meshes as well as one overset case. The largest single grid case will be compared with *Kestrel* v2.1b.

A. 3.15 Million Cell Grid

The 3.15 million cell grid is a hybrid grid around a JDAM, which is similar to the store from the F-18C store separation challenge of 1999[8]. The grid has roughly 2.7 million tetrahedrons and 430K prisms. The baseline number of PEs, or the minimum number of PEs, used on this case was 32. This baseline is used to determine the scalability on greater numbers of PEs and, typically, one tries to make the baseline as small as possible. The scale factor at the baseline will always be one. The 3.15 million cell grid was run on 32, 64, 128, 258, 512, 1024, 2048 and 4096 PEs on *raptor* using *Cobalt* V5.2. Figure 3 provides the scalability curve for this grid.

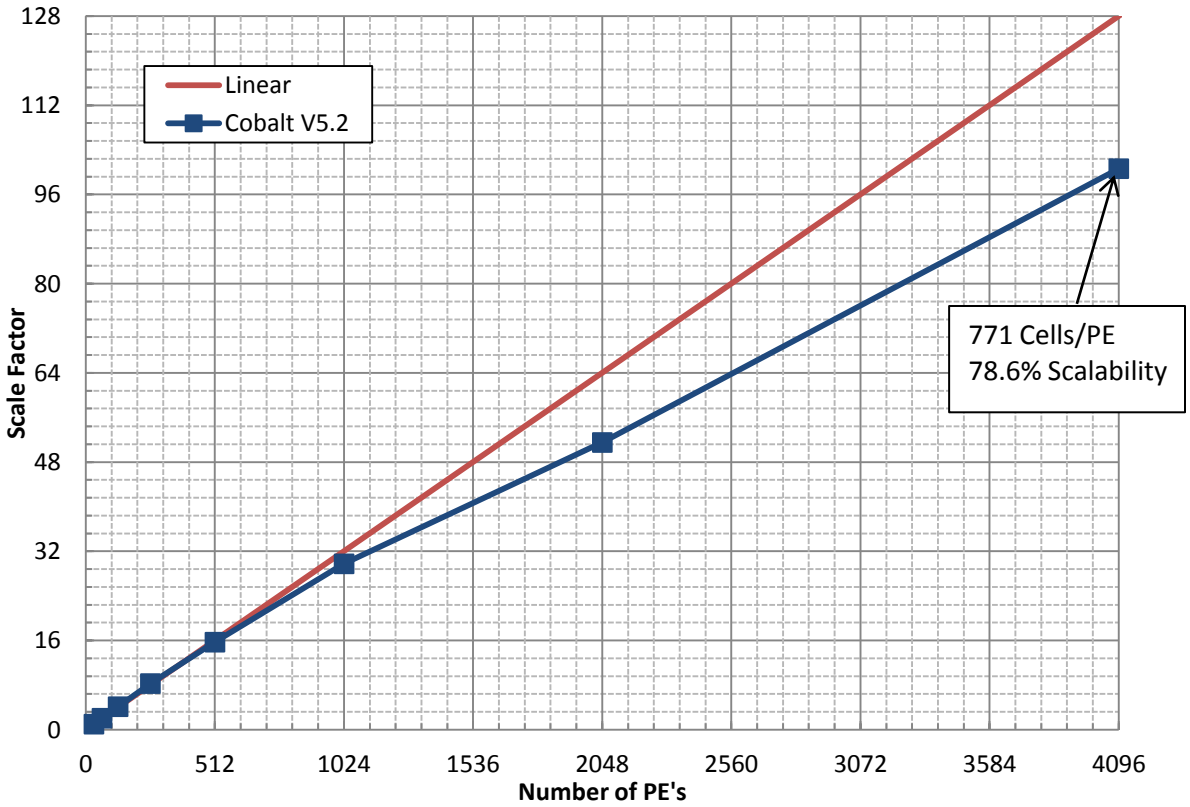

Figure 3: *Cobalt* Scalability on *Raptor* for a 3.15 Million Cell Grid

Table 1: Results for 3.15 Million Cell Grid

Number of PE's	Cells per PE	Scalability Efficiency
32	98663	100.0%
64	49332	102.9%
128	24666	103.2%
256	12333	103.5%
512	6166	98.1%
1024	3083	93.0%
2048	1542	80.5%
4096	771	78.6%

Table 1 presents scalability efficiency as a function of processor count and number of cells per processor. For this grid, *Cobalt* has super-linear speed-up for 64, 128 and 256 PEs. This is due to enhanced cache performance as the number of cells per PE decreases. Even at 771 cells per PE, *Cobalt* is achieving nearly 80% scalability efficiency. Below this number of cells per PE, communication is starting to cost more than the computation thereby decreasing scalability efficiency.

B. 10 Million Cell Grid

The 10 million cell grid is a hybrid grid around an F-18C. This grid has roughly 8 million tetrahedrons and 2.1 million prisms. The baseline number of PEs used on this case was 32. This grid contains several types of boundary conditions including no-slip solid walls, far-field, engine exhaust and inlets. The 10 million cell grid was run on 32, 64, 128, 258, 512, 1024, 2048, 4096, 8192 and 15,360 PEs on *raptor* using *Cobalt* V5.2. Figure 4 provides the scalability curve for this grid.

Table 2 shows scalability as a function of processor count and number of cells per processor. For this grid, *Cobalt* has super-linear speed-up from 64 to 2048 PEs. Even at 657 cells per PE, *Cobalt* is achieving 70% scalability efficiency.

Table 2: Results for the 10 Million Cell Grid

Number of PE's	Cells per PE	Scalability Efficiency
32	315319	100.0%
64	157659	104.4%
128	78830	106.4%
256	39415	108.3%
512	19707	111.4%
1024	9854	106.8%
2048	4927	106.3%
4096	2463	93.8%
8192	1232	89.3%
15360	657	70.3%

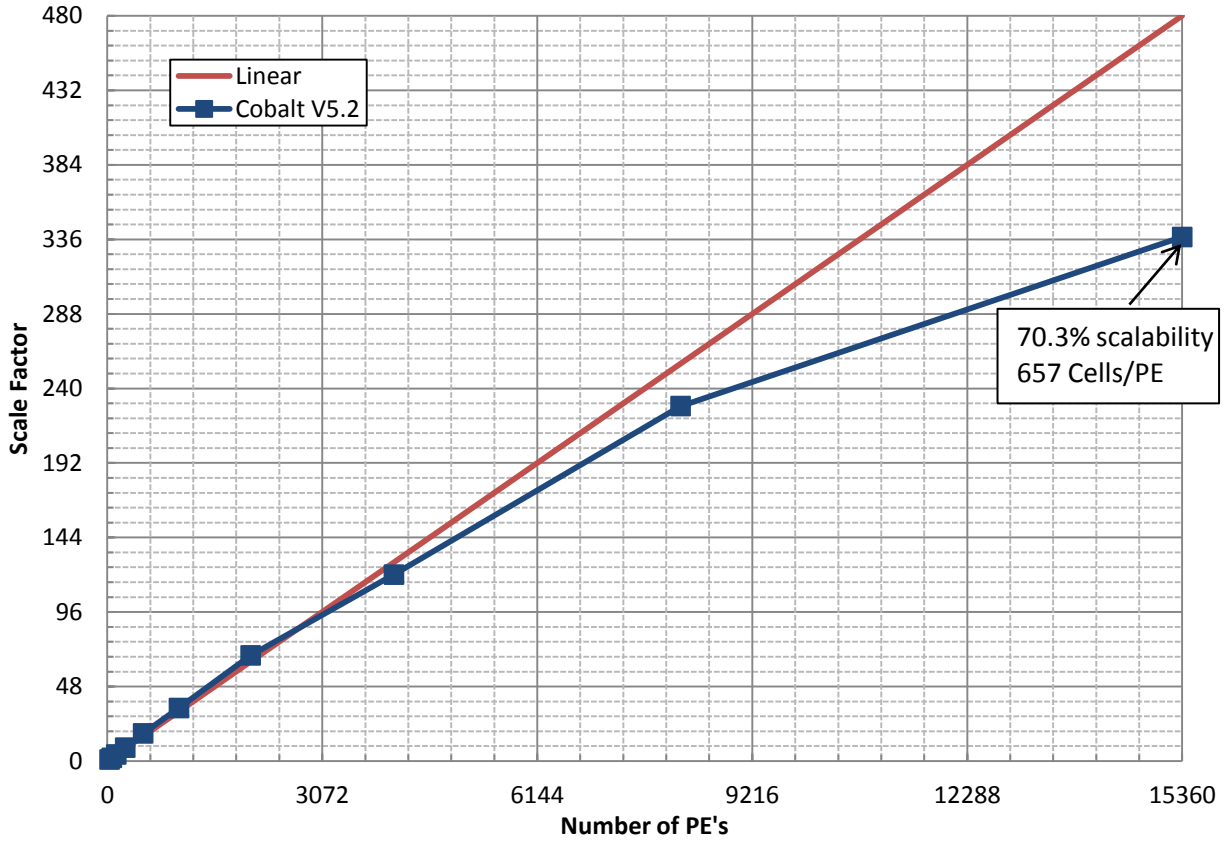

Figure 4: *Cobalt* Scalability on Raptor for a 10 Million Cell Grid

C. 49 Million Cell Grid

The 49 million cell grid is a hybrid grid around a C-17. This grid has roughly 44 million tetrahedrons and 5.3 million prisms. The baseline number of PEs used on this case was 96. This grid contains several types of boundary conditions including no-slip solid walls, far-field, engine exhaust and inlets. We selected this size of grid to be comparable to the *Kestrel* results in Figure 2. This grid was run on 96 through 21,856 PEs on *raptor* using *Cobalt* B5.11. Twenty-one thousand, eight-hundred fifty-six PEs is half the number of PEs of *raptor* and can be accessed through the regular queue. Figure 5 provides the scalability curve for this grid.

During initial scalability testing of *Cobalt* V5.2, we discovered that the grid partitioning phase began to require excessive CPU time once the number of PEs exceeded around 6000. In fact, the grid partitioning step required 50 minutes of wall clock time on 8,192 PEs. This was deemed unacceptable. Diagnosis revealed that the problem was in the integration of *Cobalt* with the ParMETIS graph partitioning software[9]. After a couple hours of rewriting code in our current beta release, we were able to reduce the grid partitioning phase to 50 seconds on 8,192 PEs and 124 seconds on 21,856 PEs. We therefore decided to use our current beta, *Cobalt* B5.11, for the scalability tests for the 49 million cell grid. This version also uses the latest release of ParMETIS, V4.0.

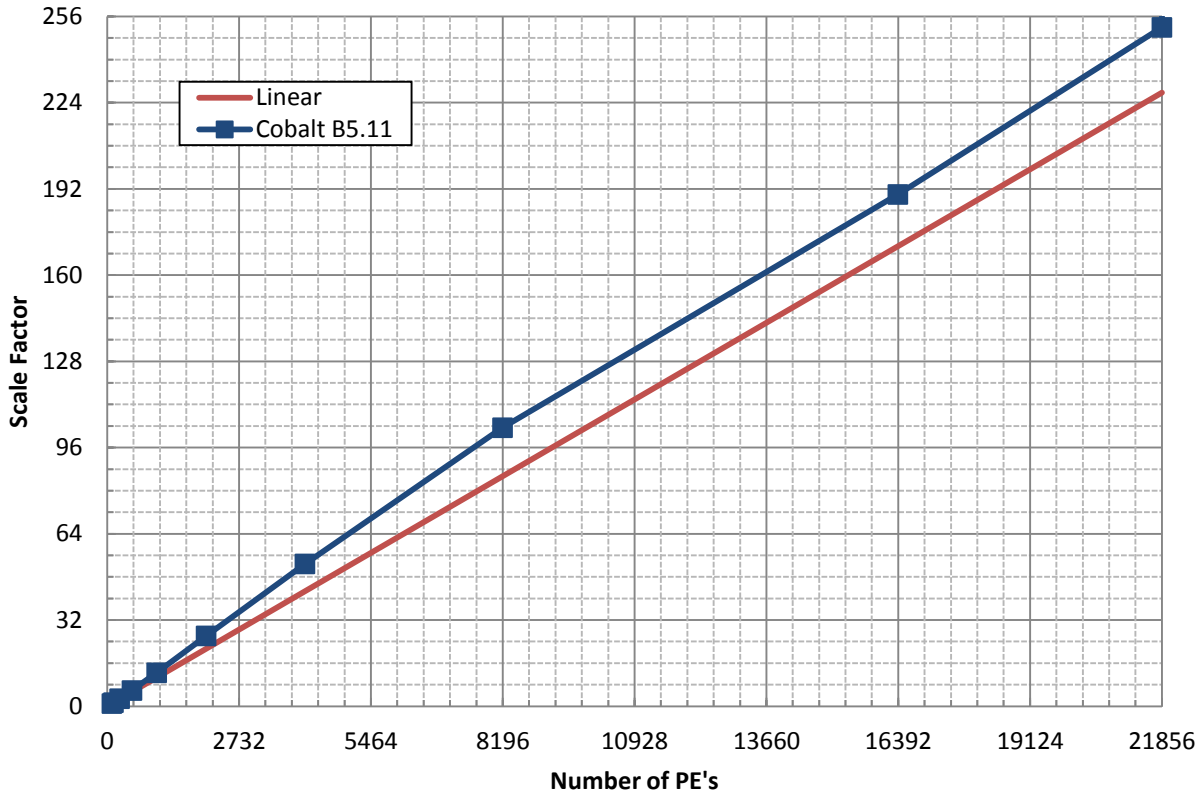

Figure 5: *Cobalt* Scalability on Raptor for a 49 Million Cell Grid

Table 3 shows scalability as a function of processor count and number of cells per processor. For this grid with this baseline, *Cobalt* has super-linear speed-up from 256 to 21,856 PEs.

Table 3: Results for the 49 Million Cell Grid

Number of PE's	Cells per PE	Scalability Efficiency
96	514662	100.0%
128	385996	100.0%
256	192998	104.9%
512	96499	109.3%
1024	48250	116.3%
2048	24125	122.4%
4096	12062	123.8%
8192	6031	121.1%
16384	3016	111.3%
21856	2261	110.7%

D. 13 Million Overset Grid

Lastly, we wanted to test the scalability of an overset case. The 13 million cell grid is a combination of the F-18C and JDAM grids using the Overset module available in *Cobalt*. The baseline number of PEs used on this case was 32. Unfortunately, each overset case will scale differently because of the hole-cuts being different for each case. The Overset module does not scale as well as a single-grid case. This case was run on 32, 64, 128, 258, 512, 1024, 2048, and 4096 PEs on *raptor* using *Cobalt* V5.2. Figure 6 provides the scalability curve for this Overset grid.

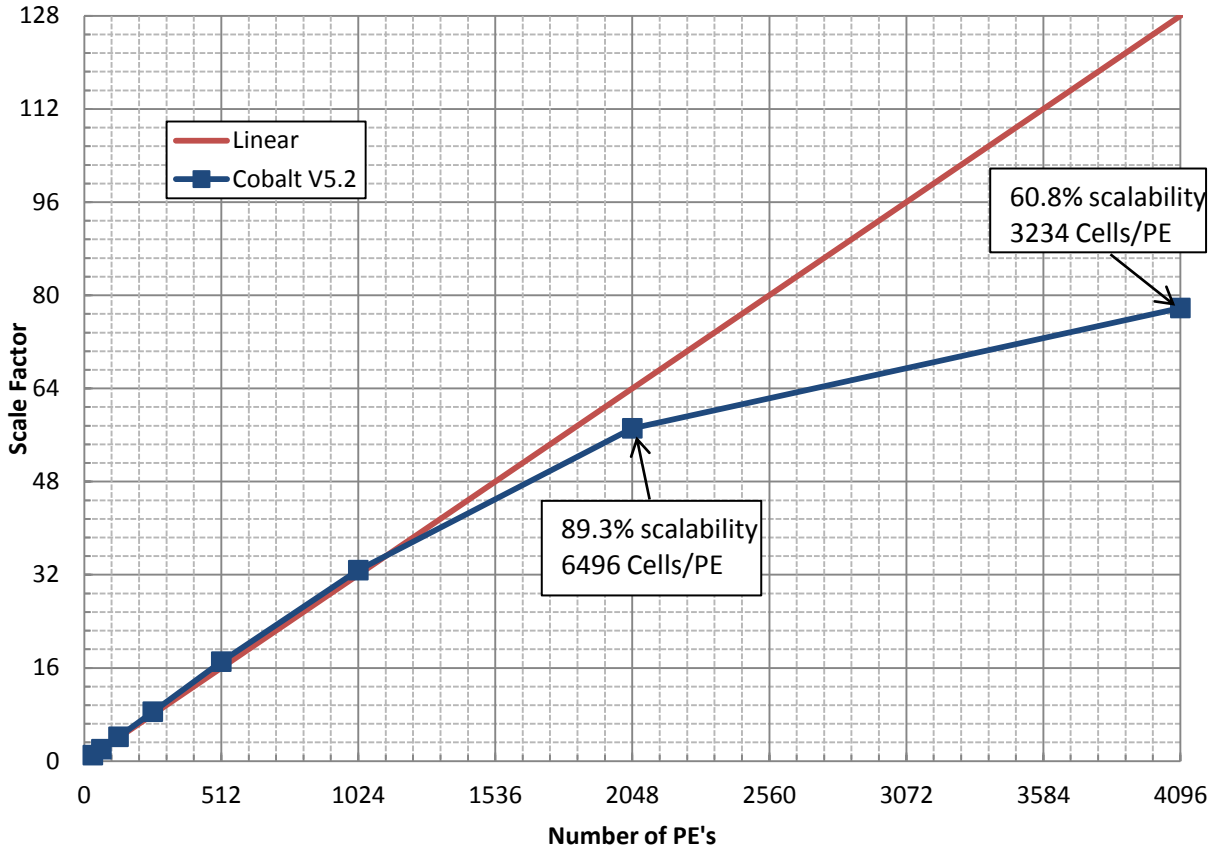

Figure 6: *Cobalt* Scability on Raptor on a 13 Million Cell Overset Grid

III. Conclusions

The goals of this study were: 1. To update the scalability charts of *Cobalt*; 2. To see if *Cobalt* still scaled well on modern machine architectures, with tens of thousands of processors; 3. To investigate the validity of the general conclusion that no legacy code can achieve this level of scalability, requiring the development of new numerics.

Figures 3, 4 and 5 show the scalability of *Cobalt* for various size grids. All three figures prove that *Cobalt* is highly scalable even on tens of thousands of processors. To compare the scalability of the three different grid sizes, scalability efficiency based on the number of cells per PE is plotted in Figure 7. We set the baseline for each grid to the data point that was closest to 100,000 cells per PE. For the 3.15 million cell grid and the 10 million cell grid, the curves are very similar. With this baseline, super-linear

speed-up is obtained when the number of cells per processor is greater than 8,000. A scalability efficiency of around 90% is achieved for 1,500 to 2,000 cells per PE or larger and 80% scalability efficiency is obtained for roughly 1,000 cells per PE. The 49 million cell grid shows even better scalability performance. This is due in part to changes in *Cobalt* from V5.2 to B5.11 and to the use of ParMETIS V4.0. Figure 7 provides some guidance for users to estimate the number of PEs they should request to obtain the best scalability for their grid.

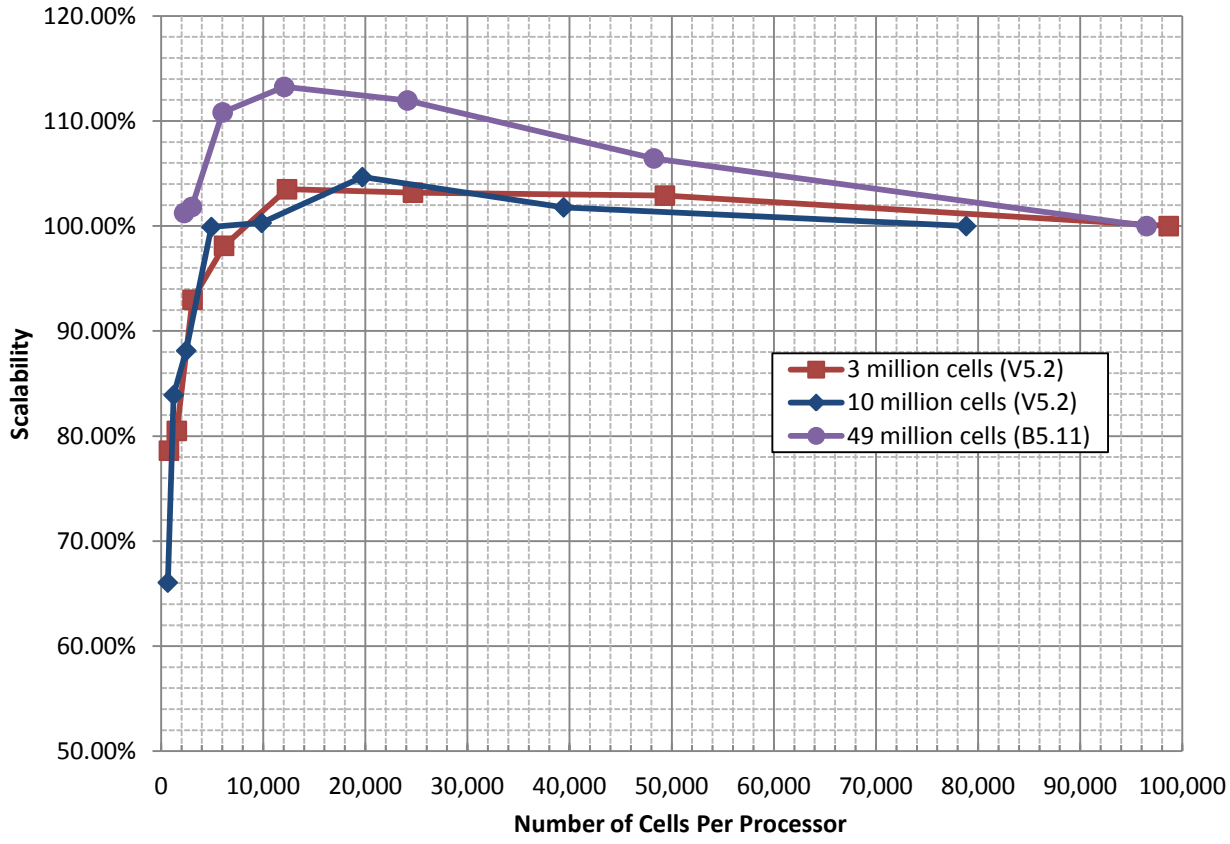

Figure 7: Scalability Efficiency versus Number of Cells Per PE

In order to compare the scalability of *Cobalt* and the current state-of-the-art CFD effort of the CREATE program, whose supposed task is to develop software that scales well on hundreds of thousands of PEs, we used a grid similar in size to their grid (49 million versus 47 million) and ran on the same machine they ran (*raptor*). To be fair, the following comparison also uses a baseline of 512 PEs, which is the value Morton, et al. used for *Kestrel* v2.1b, and not the baseline value of 96 PE's used in the *Cobalt* results shown in Figure 5. Figure 8 compares the scalability of *Cobalt* and *Kestrel*. At the maximum number of PEs that Morton, et al. presented, *Kestrel* v2.1b has 65% scalability efficiency on 4096 PEs whereas *Cobalt* achieves 113% scalability efficiency on 4096 PEs. In fact, *Cobalt* continues to be super-linear all the way up to 21,856 PEs which is the maximum number we used. Further, *Kestrel* has a scalability efficiency of 65% with 11,475 cells/PE. The lowest scalability efficiency of *Cobalt* found in

this study is 70.3% with 657 cells/PE. Using Tables 1-3 and Figure 7, we estimate that *Cobalt* would have a scalability efficiency greater than 80% on 60,000 PEs with this 49 million cell grid.

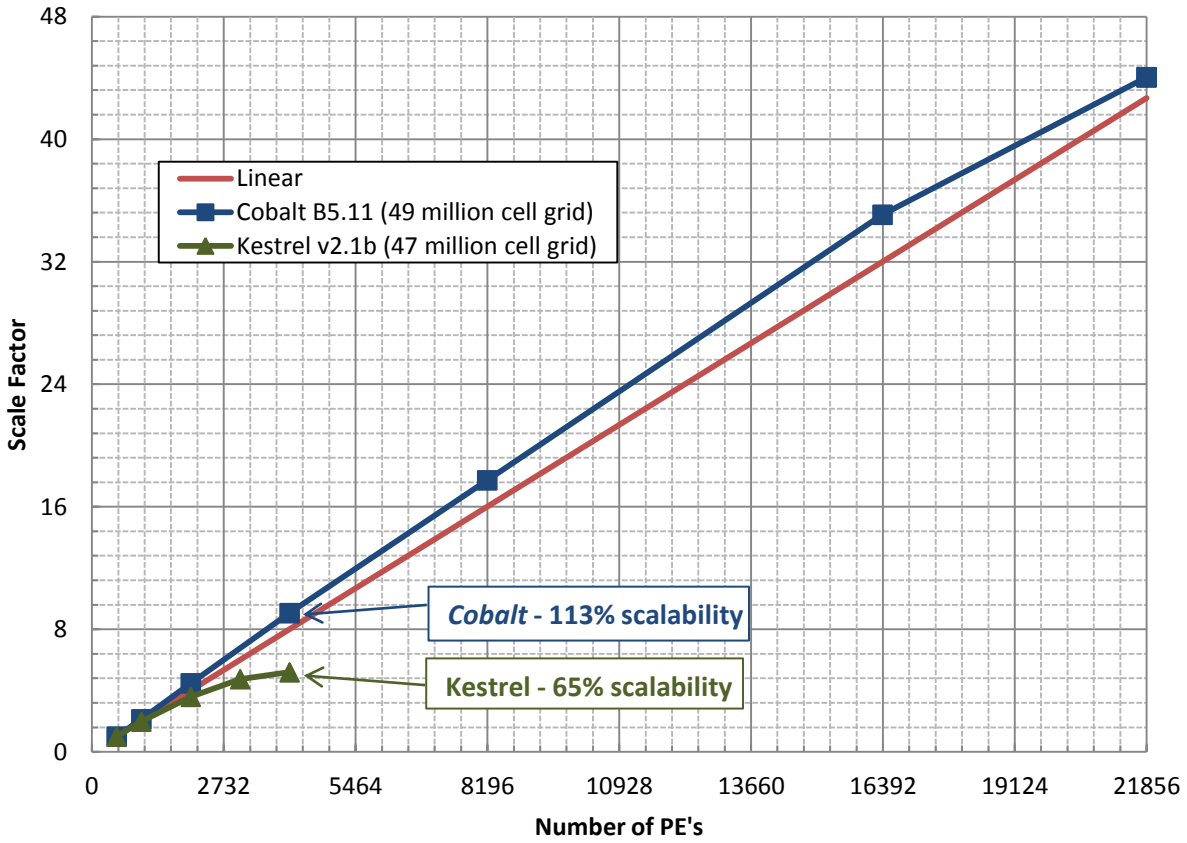

Figure 8: Comparison of *Cobalt* and *Kestrel* Scalability on Raptor

The current versions of *Cobalt*, V5.2 and B5.11, both available on most HPC machines, have proven to be highly scalable, scaling well on tens of thousands of processors. We have also shown that *Cobalt* clearly has the potential to scale well on hundreds of thousands of processors, given a grid size of roughly 85 million cells or larger. We have shown that scalability efficiency for *Cobalt* can be roughly predicted based on a cells-per-PE basis. Finally, the general conclusion that so-called legacy codes cannot scale well on tens to hundreds of thousands of processors has been shown to be erroneous.

References

[1]Strang, W. Z., Tomaro, R. F., and Grismer, M. J., "The Defining Methods of Cobalt: A Parallel, Implicit, Unstructured Euler/Navier-Stokes Flow Solver," AIAA Paper 1999-0786, 1999.
[2]Tomaro, R. F., Strang, W. Z., and Sankar, L. N., "An Implicit Algorithm For Solving Time Dependent Flows on Unstructured Grids," AIAA Paper 1997-0333, 1997.
[3]*Navigator*, NAVO MSRC, Spring 2005.

[4]*Wright Cycles*, ASR, Spring 2006.
[5]ERDC MSRC *Resource*, Fall 2005.
[6]Morton, S. A., Eymann, T. A., Lamberson, S. R., McDaniel, D. R., Sears, D. R., Tuckey, T. R. and Utrilla, J., *Rigid and Maneuvering Aircraft Results for Kestrel v2 with Kestrel v3 Design Attributes*, DoD HPCMP User's Group Conference, Portland OR, 21-23 June 2011.
[7]Douglas, C. C. and Zornes, A. F., "Computational Fluid Dynamics (CFD) Modeling and Analysis, Delivery Order 0006: Cache-Aware Air Vehicles Unstructured Solver (AVUS)," AFRL-VA-WP-TM-2006-3009, August, 2005.
[8]Tomaro, R.F, Witzeman, F. C. and Strang, W. Z., "A Solution on the F-18C for Store Separation Simulation Using $Cobalt_{60}$," AIAA Paper 1999-0122
[9]Karypis, G. and Kumar, V., "Multilevel Algorithms for Multi-Constraint Graph Partitioning," Supercomputing, May 5, 1998.