Transparent Grid Enablement of WRF Using a Profiling, Code Inspection, and Modeling Approach

Team: S. Masoud Sadjadi[1], Javier Muñoz[1], Diego Lopez[1], Javier Figueroa[1], Xabriel J. Collazo-Mojica[6], David Villegas[1], Raju Rangaswami[1], Shu Shimizu[4], Hector A. Duran Limon[5], Rosa Badia[2], Pat Welsh[3], Jason Liu[1], Alex Orta[1], Michael McFail[1], and Erik Jones[3]

1: Florida International University, Miami, FL, USA; ***2:*** Barcelona Supercomputing Center, Barcelona, Spain; ***3:*** University of North Florida, Jacksonville, FL, USA; ***4:*** IBM Tokyo Research Laboratory, Tokyo, Japan; ***5:*** University of Guadalajara, Mexico; ***6:*** University of Puerto Rico- Mayaguez Campus, Puerto Rico;

I. Motivation

- **The impact of the hurricanes is so devastating** throughout different levels of the society that there is a pressing need to provide a range of users with accurate, timely information to enable effective planning for and response to potential hurricane landfalls.
- **The popular Weather Research and Forecasting (WRF) model** is the latest numerical model developed by the National Center for Atmospheric Research (NCAR) for both operational forecasting and atmospheric research and has been adopted by meteorological services in the U.S. and world wide.
- **The high resource requirements of the WRF** demand a large number of computing nodes with high volume of memory and storage, connected through high-speed networks.
- **The budget limitation** is the main inhibiting factor that prevents typical organization from satisfying the increasing resource requirements of the WRF code! Unfortunately, the WRF code was not developed to scale out to a Grid computing environment.
  - Note: the current version of WRF is designated to run either on a single machine or on a cluster of homogeneous nodes.

II. Goals

- **Enabling WRF to scale out to Grid computing environments** so that it can benefit from the available resources in other partner organizations.
- **Modeling WRF behavior and its resource requirements** to estimate the time required for a simulation given a particular set of resources and predict the allocation of resources.
  - For example, the optimized number of homogenous nodes required for a hurricane path prediction simulation.

This poster focuses on this goal.

III. Challenges

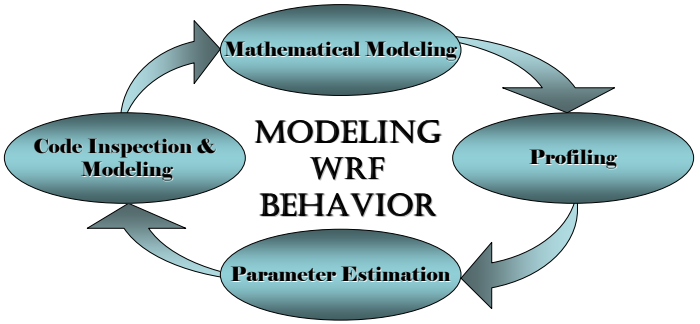
- **The high latency of Internet compared to high-speed LANs** does not satisfy the real-time requirements of the WRF code.
  - For example, we cannot simply use domain decomposition for grid enablement of WRF due to significant overhead as the boundary grid points assigned to run on the resources in one organization need to communicate with their neighboring grid points assigned to another partner organization potentially across the Internet.
- **The high overhead of the Grid middleware software**, such as Globus Toolkit (GT4) and Community Toolkit (CoG), and inefficiency of the meta-schedulers are other inhibitors for Grid-enablement of real-time applications like WRF.
- **Risking compatibility with future WRF versions**, if we make manual and ad-hoc modifications to the WRF source code. Therefore, we need a systematic approach for Grid enablement of the WRF code, where the modification is transparent to the original code.
- **The high volume of the WRF sources code** makes it hard to get a full grasp of the whole code and to model its behavior.
  - ~ 165,000 lines of source code and another 40,000 lines generated at compile time.
- **Compiling WRF on unsupported platforms** is a tedious task! NCAR has supported several platforms, but unfortunately NCAR has not yet provided configuration files to make the compilation of WRF straightforward on some of the platforms at our disposal.
  - For example, the Power5 cluster at UNF and the Power4 31-way node at FIU.

IV. Mathematical Modelling

- We assume that
  - resource consumption consists of any forms of static resource parameters such as CPU cycles.
  - parallelism (the 1st bracket term in the mathematical model below) is independent of the static resource parameters (the 2nd bracket term).
- The expression is finally transformed into a linear summation of profile parameters (θ's), which are estimated as an application's characteristic.

$$x = \left(\alpha_0 + \sum_{k=1}^{4} \alpha_k n^k\right)\left(\beta_0 + \beta_1 / b_{\text{CPU}} + \beta_2 / b_{\text{cache}} + \beta_3 b_{\text{memory}} + \beta_4 b_{\text{disk}} + \beta_4 b_{\text{network}} \Lambda\right) + \varepsilon$$
$$\equiv \theta_0 + \sum_{k=1}^{4} \theta_k y_k + \varepsilon$$

x: Criterion variable (e.g., execution time)
n, b_k: Parallerism and static resource parameters (given)
y_k: Explanatory variables (combinati on of n and b_k)
$\alpha_k, \beta_k, \theta_k$: Profile parameters (to be estimated)
ε: Error



V. Profiling

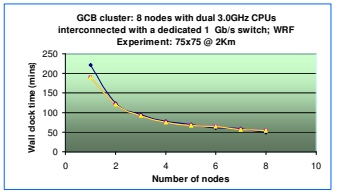
- We evaluated 21 existing profilers
  - Vampir/ITAC, XMPI, MPIP, KOJAK, Paraver, Gprof, TAU, PGI CDK, OPT, MPE/Jumpshot, Paradyn, SvPablo, VTune, HPC Toolkit, Etnus TotalView, OpenISpeedShop, Oprofile, PapiEx, IPM, DEEP/MPI, and PerfSuite.
  - We chose to use IBM Toolkit, Paraver, and TAU for our profiling experiments.

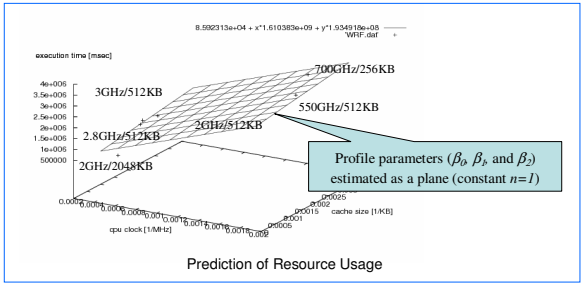
TAU	
Pros: - It has a profile visualization tool that provides graphical displays of all the performance analysis results, in aggregate and single node/context/thread forms. - It can generate event traces that can be displayed with the Vampir, Paraver or JumpShot trace visualization tools. - It supports MPI and OpenMP. It will soon support g95. - Free license, just have to request a copy through email. - Good documentation.	**Cons:** - No g95 Support yet! - Instrumentation is tedious and error prone. - Steep learning curve. - Many features depend on other software.

- We developed custom-designed profiling tools
  - **amon** is a monitoring tool that collects static resource parameters, including CPU speed and cache size, and provides the average dynamic resource consumption, including cpu time, memory, network, and storage.

VI. Parameter Estimation

- Application profiles are derived by executing the application on different platforms with varied configuration of available resources.
- Regression analysis is then used to fit the data into a linear model (in terms of profile parameters).
- As more observations are made, the accuracy of the model generated improves.
- This model is then subsequently used for predicting application execution and resource usage on previously "unseen" platforms.

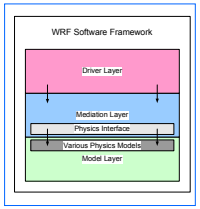


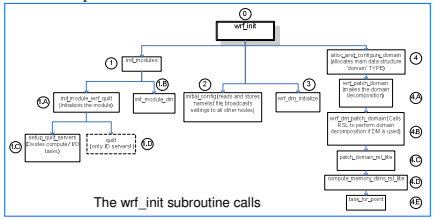


Prediction of Resource Usage

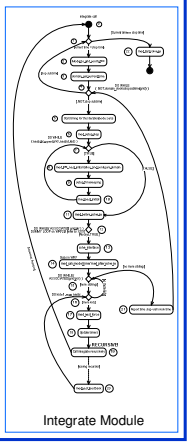
VII. Code Inspection & Modeling

- We use code inspection and modelling to justify why WRF behaves as it does.
- We provide feedback to the mathematical modelling
  - may result in adding or removing parameters
  - may result in reflecting the dependencies of two or more parameters.





The wrf_init subroutine calls



Integrate Module