# A High Performance Computing Study of a Scalable FISST-Based Approach to Multi-Target, Multi-Sensor Tracking

**Islam Hussein, Matthew P. Wilkins, and Christopher W. T. Roscoe**
*Applied Defense Solutions*
**Weston Faber and Suman Chakravorty**
*Texas A&M University*
**Paul W. Schumacher**
*United States Air Force Research Laboratory*

## ABSTRACT

Finite Set Statistics (FISST) is a rigorous Bayesian multi-hypothesis management tool for the joint detection, classification and tracking of multi-sensor, multi-object systems. Implicit within the approach are solutions to the data association and target label-tracking problems. The full FISST filtering equations, however, are intractable. While FISST-based methods such as the PHD and CPHD filters are tractable, they require heavy moment approximations to the full FISST equations that result in a significant loss of information contained in the collected data. In this paper, we review Smart Sampling Markov Chain Monte Carlo (SSMCMC) that enables FISST to be tractable while avoiding moment approximations. We study the effect of tuning key SSMCMC parameters on tracking quality and computation time. The study is performed on a representative space object catalog with varying numbers of RSOs. The solution is implemented in the Scala computing language at the Maui High Performance Computing Center (MHPCC) facility.

## 1. INTRODUCTION

The advancing technologies utilized by space observing systems improve our ability to observe and track the over 500,000 resident space objects (RSOs) of 1 cm size or greater. This improvement in capability is immensely beneficial to Space Situational Awareness (SSA) but also introduces new challenges to the execution of SSA's underlying tasks. Such tasks include the detection, tracking, identification and characterization of the very large number of objects in space, along with an ability to assess conjunctions and detect and identify threats. This presents the space community with a very challenging computational problem given the large number of objects to be catalogued, the large number of parameters needed to completely characterize each RSO, and the amount of observations returned from space observing assets. While the goal of the work included in this paper is aimed at addressing this overall computational problem in SSA, of specific interest in this paper is to efficiently process raw sensor observations. Efficiency here is in terms of both computational efficiency (i.e., real-time processing of the data) and informational efficiency (i.e., extracting the most amount of information from the collected data).

Towards that end, the authors have developed The Joint Centralized Autonomous Tasking System (JCATS), which is a new and innovative tool aimed at addressing several needs of the SSA tasks. For more on the JCATS system, please refer to Ref. [1]. JCATS implements new analysis tools and techniques to automate many aspects of the SSA workflow as much as possible and improve the effective use of available resources. First, two new tools, leverage from previous AFRL efforts, were implemented as the core analysis engine for JCATS. A data fusion engine based on Finite Set Statistics (FISST) [2, 3] is used to take observations and maintain an RSO catalog. Due to the multi-hypothesis nature of FISST this allows for various orbit events, such as maneuvers, to be analyzed and considered during observation processing. The Information State Receding Horizon Control (ISRHC) tool [4] is used to optimize a surveillance system tasking schedule based on information gain. Secondly, a big innovation that JCATS has implemented is the use of information-based metrics for analysis. Instead of specifying that an object be observed at certain average frequency the observation schedule is determined by the requirements set, such as a threshold position uncertainty level or the proximity to another object. As these requirements are calculated they raise or lower the effective priority of an object, which in turn influence the amount of information gain from a set of observations. Thirdly, JCATS' approach to the SSA problem offers alternative standard operating procedures to maximize effectiveness. The transfer of data and instructions between components is primarily driven through a net-centric, service-oriented architecture (SOA). This allows for a design that can provide rapid response capabilities that adapt to real-time circumstances.

In this paper, we focus primarily on the FISST-based observation ingestion and filtering component of JCATS and relegate the discussion of the other components of JCATS to another paper [1]. FISST is capable of handling phenomena such as clutter, misdetections, and object birth and death. Implicit within the approach are solutions to the data association and target label-tracking problems. Even for problems without object birth and death, i.e., just accounting for data association challenges and the associated misdetection and clutter errors, the full FISST filtering equations are intractable. While FISST-based methods such as the PHD and CPHD filters are tractable, they require heavy moment approximations to the full FISST equations that result in a significant loss of information contained in the collected sensor data. Recently, the authors have developed an approximation of the full FISST equations. These approximations do not require any moment approximations, hence retaining much of the information content of sensor data, yet they render the full FISST equations tractable and scalable. The key idea behind these approximations lies in the update step of the filter. While the prediction step is embarrassingly parallelizable, the update step is partially parallelizable. The bottleneck computation in the full FISST update step is the requirement to enumerate all possible observation-to-track associations.

The core idea in rendering the FISST filter tractable is that instead of considering all associations, one can include only a subset of the associations. If the chosen subset contains the true association, only a small amount of irrelevant information is lost. If the true association is not among the chosen subset, while some tracking error will be introduced, the update step becomes essentially trivial in that the filter only weakly adjusts the posterior multi-target probability density function. The subset of associations can be chosen in any one of various ways. For example, they can be selected uniformly randomly. Alternatively, one can select them according to a Markov Chain Monte Carlo (MCMC) technique partially developed by the authors, called Smart Sampling MCMC (SSMCMC). In SSMCMC one sets the number of selected associations $M$ for inclusion in the update step and the method returns a subset of the associations of size $M$. In the next section we will review the FISST approximation and in a later section we will review the SSMCMC approach. We will then study the effect of tuning the SSMCMC parameter $M$, along with other filter parameters, on tracking quality. The study will be performed on a representative space object catalog with varying numbers of RSOs. We will implement the solution using the Scala computing language, which naturally lends itself to parallelism and distribution. The study was conducted on AFRL's high performance computing facility. We will demonstrate the scalability of the developed solution as a function of the number of RSOs.

## 2. FINITE SET STATISTICS FOR MULTI-TARGET, MULTI-SENSOR FILTERING

FISST is a hybrid Bayesian filtering technique that updates any available prior information state (described using a multi-target, multi-hypothesis probability density function (PDF)) using new incoming sensor measurement observations, discrete and non-traditional observations, potentially soft data and the available sensor phenomenology. At its heart, FISST is a rigorous hypothesis management filter. FISST has many capabilities including rigorous handling of process and measurement noise, false alarm hypothesis tracking (non-zero probability of detecting non-objects) and misdetection hypotheses (i.e., probability of detection less than 1), object correlation and observation association and associated errors, object birth (e.g., delamination, new launches, and object spawning and breakup events), death (e.g., re-entry), and classification/characterization hypotheses (e.g., active/inactive, HAMR/non-HAMR, object function, maneuverable/non-maneuverable, etc.). Capabilities such as object birth, object spawning, object characterization and classification are critical to many SSA applications. These capabilities are what enables FISST to capture any and all informational dependence between object motion tracking (i.e., catalog processing) and discrete hypotheses (e.g., for RSO characterization).

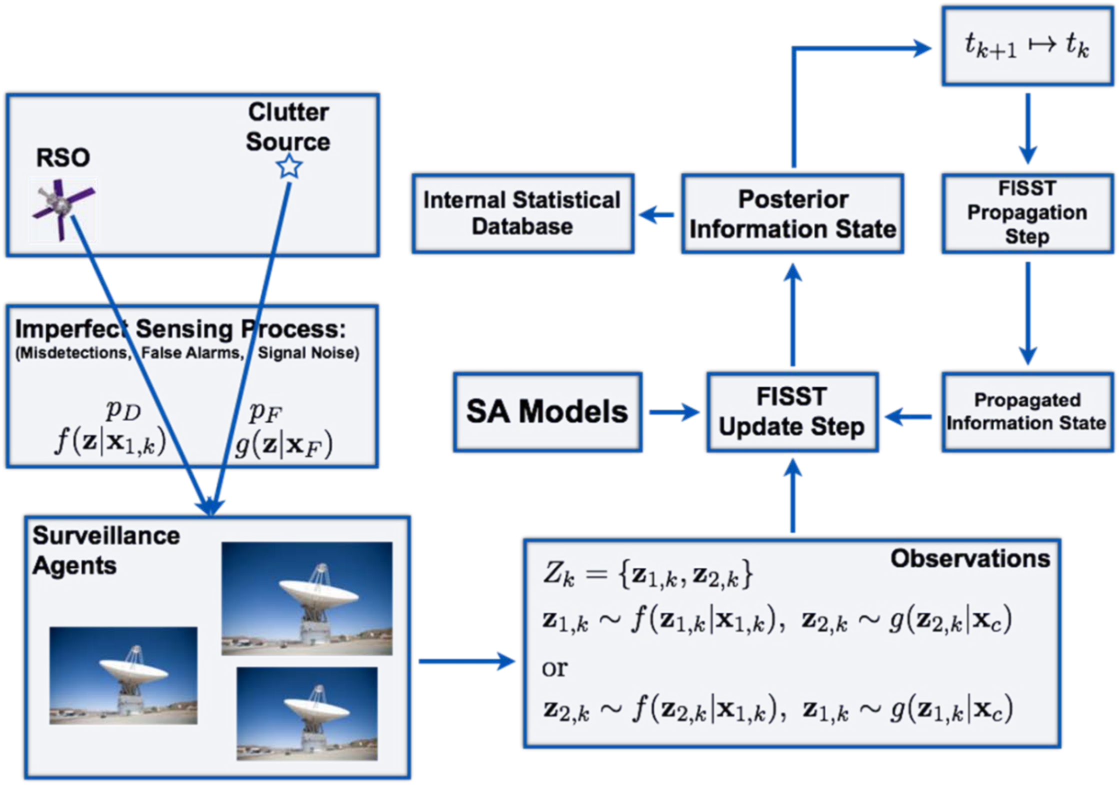
During periods when no new collections are received, FISST propagates this information state using the available multi-hypothesis propagation models. Such models incorporate, for example, satellite motion propagation. This propagation step results in increased uncertainty in the RSO information state. The RSO gets a reduction in uncertainty when new observations arrive to update its state. Mathematically this is done by recursively solving the following two equations [2]:

$$f_{k|k-1}\left(X_k \middle| Z^{(k-1)}\right) = \int f_{k|k-1}(X_k|X'_{k-1}) f_{k-1|k-1}\left(X'_{k-1} \middle| Z^{(k-1)}\right) \delta X'_{k-1} \tag{1}$$

$$f_{k|k}(X_k|Z_k) = \frac{f_k(Z_k|X_k) f_{k|k-1}\left(X_k \middle| Z^{(k-1)}\right)}{f_k(Z_k|Z^{(k-1)})} \tag{2}$$

where the function $f_{k-1|k-1}\left(X'_{k-1} \middle| Z^{(k-1)}\right)$ is the prior multi-target FISST-based hybrid PDF, $f_{k|k-1}(X_k|X'_{k-1})$ is the multi-target Markov transitional PDF, $f_{k|k-1}\left(X_k \middle| Z^{(k-1)}\right)$ is the propagated prior multi-target PDF, $f_k(Z_k|X_k)$ is the multi-target likelihood function that describes the likelihood of getting a measurement $Z_k$ given the true state is $X_k$, and, finally, $f_{k|k}(X_k|Z_k)$ is the multi-target posterior PDF given a new measurement $Z_k$. A FISST-based hybrid PDF is one that integrates the notion of discrete probability mass functions and continuous-state probability density functions in one holistic and rigorous probabilistic framework. The integral in Eq. (2) is a set integral [5] and reduces to a sum over all discrete hypotheses of the state $X_k$, and for each discrete hypothesis a continuous integral over the continuous part of the state $X_k$ evaluated at that hypothesis. This notion of integration ensures that FISST is a rigorous mathematical framework for hybrid estimation problems. It is this integral that assures us that any probabilistic coupling between continuous and discrete states gets captured in the analysis, resulting in the extraction of the most amount of information from the provided heterogeneous raw data. Accordingly, as a Bayesian technique, FISST is continuously executing two basic operations as shown in Fig. 1. The first is to update the internal database at observation collection times. This step usually results in a reduction in uncertainty in the observed object's state. In between observations, the internal statistical database is propagated (i.e., uncertainty propagation) according to established astrodynamic propagation models. This step usually results in increased uncertainty in the RSO information state.

FISST is, essentially, a rigorous hypothesis management technique. Hypotheses it accounts for include: object birth and death, data association hypotheses, misdetection and false alarm hypotheses, as well as object classification hypotheses. Given a hypothesis, FISST uses any of the existing uncertainty propagators, such as the Unscented Kalman Filter (UKF) or the Particle Filter (PF) to propagate and update the underlying RSO information states for that hypothesis. JCATS currently is capable of using either the UKF or the PF, or an intelligent combination of the two for maximal computational efficiency, to process the hypotheses. Unlike ad hoc techniques such as Multiple Hypothesis Tracking (MHT), FISST is mathematically rigorous which enables it to take into account that all such hypotheses are statistically dependent. FISST is capable of using such dependencies to further assert or reject hypotheses based on incoming observations. This allows FISST to extract the most amount of information from the data.



$t_{k+1} \mapsto t_k$
Clutter Source
RSO
Internal Statistical Database
Posterior Information State
FISST Propagation Step
Imperfect Sensing Process:
(Misdetections, False Alarms, Signal Noise)
$p_D$
$f(\mathbf{z}|\mathbf{x}_{1,k})$
$p_F$
$g(\mathbf{z}|\mathbf{x}_F)$
SA Models
FISST Update Step
Propagated Information State
Surveillance Agents
Observations
$Z_k = \{\mathbf{z}_{1,k}, \mathbf{z}_{2,k}\}$
$\mathbf{z}_{1,k} \sim f(\mathbf{z}_{1,k}|\mathbf{x}_{1,k}),\ \mathbf{z}_{2,k} \sim g(\mathbf{z}_{2,k}|\mathbf{x}_c)$
or
$\mathbf{z}_{2,k} \sim f(\mathbf{z}_{2,k}|\mathbf{x}_{1,k}),\ \mathbf{z}_{1,k} \sim g(\mathbf{z}_{1,k}|\mathbf{x}_c)$

**Figure 1: FISST's Internal Information Flow**

The general filtering problem (Eq. (1) and Eq. (2)) is computationally intractable. FISST-based children techniques such as the Probability Hypothesis Density (PHD) and the Cardinalized PHD (CPHD) [2], and to a lesser extent the δ-GLMB [6], involve heavy moment approximations to FISST to make it tractable. However, these solutions result in severe loss of information content. The authors have led an effort over the last three years attempting to make solving Eq. (1) and Eq. (2) tractable without resorting to moment approximation techniques. The result was a FISST-based JCATS solution that was demonstrated to be tractable and scalable with respect to the amount of raw observational data, the number of RSOs in a space environment and the number of sensors collecting data [1]. The core idea behind the scalability of the proposed solution is that the discrete hypotheses are filtered according to an intelligent implementation of Markov Chain Monte Carlo (MCMC) sampling, called Smart Sampling MCMC (SSMCMC) that is described and demonstrated in [7] and that will be reviewed briefly in the next section. While SSMCMC is an approximation technique, it results in a far smaller amount of information loss than those resulting from the moment approximations involved in the PHD and CPHD methods. This theoretical statement regarding information retention is currently being tested and verified by the authors. With this SSMCMC implementation, 100% of the propagation equation Eq. (1) is embarrassingly parallelizable, while the computational bottleneck lies in Eq. (2), specifically in the computation of the denominator in Eq. (2), while the numerator is completely computationally parallelizable. We note here that FISST solves the track and observation correlation problems intrinsically which is generally intractable. While SSMCMC can be used to manage all types of hypotheses (birth/death, number of objects, data association, maneuvering, etc.) in FISST, in this paper we focus only on hypotheses generated from ambiguities in data association. We review SSMCMC next.

## 3. AN SSMCMC APPROACH TO DATA ASSOCIATION FOR FISST TRACTABILITY

SSMCMC is a technique initially developed to address the data association problem inherent to multi-object tracking and later expanded to address more general types of hypotheses such as object birth [8, 9]. In scenarios where the number of possible hypotheses is too large to generate exhaustively, SSMCMC keeps the problem computationally tractable. It does this by creating a Markov Chain using the hypotheses as states of the chain. Walking through these states in an MCMC fashion, SSMCMC will sample the correct and most probable hypotheses without having to generate all possible hypotheses. Initially, SSMCMC was used to sample the top hypotheses in scenarios where the number of objects was fixed [8]. SSMCMC has been adapted to include multiple birth and multiple death hypotheses [9]. Implementation of SSMCMC works as follows:

Firstly, one starts by generating a matrix of the individual object to measurement likelihoods (Figure 2).

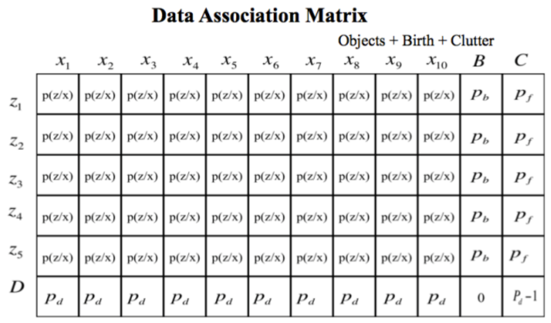
Next, one randomly generates a hypothesis by randomly selecting measurement to object pairs (note: one may use a starting hypothesis based on say the Global Nearest Neighbor rule). This hypothesis is the "Current Hypothesis" or the current location in the Markov Chain (Figure 3).

One then creates another hypothesis by switching one association by randomly selecting one measurement return and one object or association to assign it to. This hypothesis is the "Proposed hypothesis" or the state of the Markov chain you consider walking to (Figure 4).

Both the current hypothesis and the proposed hypothesis must be valid hypotheses, meaning that the hypotheses may not contain associations where multiple measurements are assigned to the same object or multiple objects are assigned to the same measurement.

One then compares the two hypotheses using an MCMC criterion that favors the more probable hypothesis. In other words, calculate the probability of both hypotheses, if the proposed hypothesis has a higher probability, then use that hypothesis as the current hypothesis. If the proposed hypothesis has a lower probability then determine whether or not to use it as the current hypothesis with probability proportional to the ratio of the hypothesis probabilities.

Repeat these steps until the assumed stationary distribution is reached. Then continue walking to sample the hypotheses in that distribution. We note here that valid and likely hypotheses are added to a collection of hypotheses to be included in the FISST update step (Eq. (2)). This collection has a maximum of M hypotheses. If $M$ is equal to all possible hypothesis combinations, we get the exact FISST update step, which is generally intractable. The iteration is stopped if a maximum number of iterations $L$ is reached or if the collection of $M$ hypotheses is unchanged for $C$ consecutive iterations (i.e., stationarity).

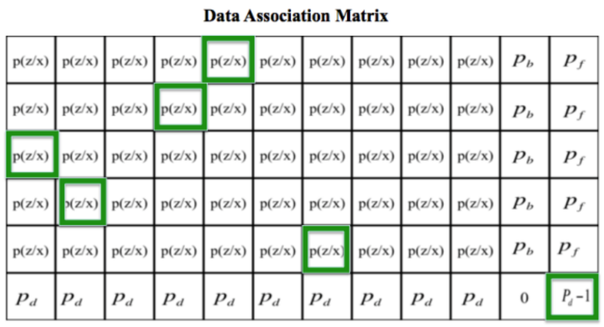


**Data Association Matrix**

Objects + Birth + Clutter

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ | $B$ | $C$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $z_1$ | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | $P_b$ | $P_f$ |
| $z_2$ | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | $P_b$ | $P_f$ |
| $z_3$ | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | $P_b$ | $P_f$ |
| $z_4$ | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | $P_b$ | $P_f$ |
| $z_5$ | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | $P_b$ | $P_f$ |
| $D$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | 0 | $P_d-1$ |

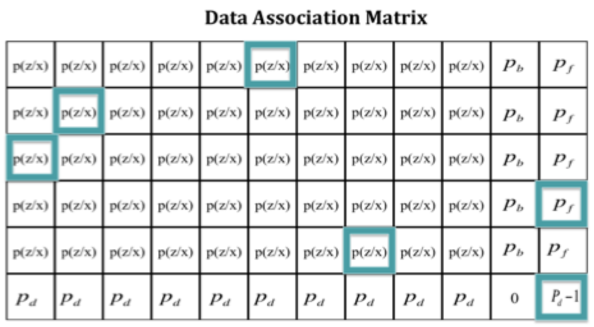
**Figure 2: The initial matrix**



**Data Association Matrix**

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| p(z/x) | p(z/x) | p(z/x) | p(z/x) | $\boxed{p(z/x)}$ | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | $P_b$ | $P_f$ |
| p(z/x) | p(z/x) | p(z/x) | $\boxed{p(z/x)}$ | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | $P_b$ | $P_f$ |
| $\boxed{p(z/x)}$ | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | $P_b$ | $P_f$ |
| p(z/x) | $\boxed{p(z/x)}$ | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | $P_b$ | $P_f$ |
| p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | $\boxed{p(z/x)}$ | p(z/x) | p(z/x) | p(z/x) | $P_b$ | $P_f$ |
| $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | 0 | $\boxed{P_d-1}$ |

**Figure 3: The current hypothesis**



**Data Association Matrix**

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | $\boxed{p(z/x)}$ | p(z/x) | p(z/x) | p(z/x) | p(z/x) | $P_b$ | $P_f$ |
| p(z/x) | $\boxed{p(z/x)}$ | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | $P_b$ | $P_f$ |
| $\boxed{p(z/x)}$ | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | $P_b$ | $P_f$ |
| p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | $P_b$ | $\boxed{P_f}$ |
| p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | p(z/x) | $\boxed{p(z/x)}$ | p(z/x) | p(z/x) | $P_b$ | $P_f$ |
| $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | $P_d$ | 0 | $\boxed{P_d-1}$ |

**Figure 4: The proposed hypothesis**

Note that the figures depicted on the right are for the SSMCMC with multiple birth and death. The process is the same for a fixed number of RSOs, but only consider Columns 1-10 and Rows 1-5.

Figure 5 depicts an SSA multi-object tracking scenario where the number of possible hypotheses is $\mathcal{O}(10^{11})$ even with a standard gating technique. Exhaustive generation of hypotheses is computationally intractable on most standard platforms. It is for scenarios such as these that SSMCMC is particularly suited.

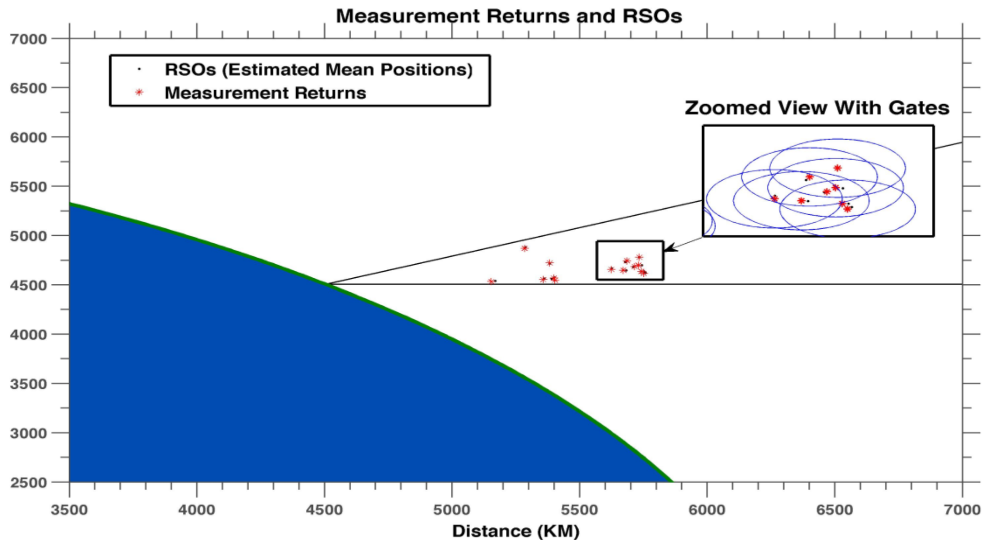


**Figure 5: An SSA Application with $\mathcal{O}(10^{11})$ hypotheses**

Figure 6 highlights the benefits of SSMCMC by showing a comparison of two different tracking methods. The blue line shows the first method called Hypothesis Oriented Multiple Hypothesis Tracking (HOMHT) method. This technique uses an exhaustive hypothesis generation technique. The second method, represented by the green line, is the Randomized FISST technique, which uses SSMCMC. It can be seen from the figure that as the number of possible hypotheses grows the computational burden caused by the exhaustive technique grows exponentially and eventually fails once the number of possible hypotheses reaches the size of the platforms memory limits. The SSMCMC technique however shows very slow growth and continues to perform far past the limitations of the exhaustive technique.

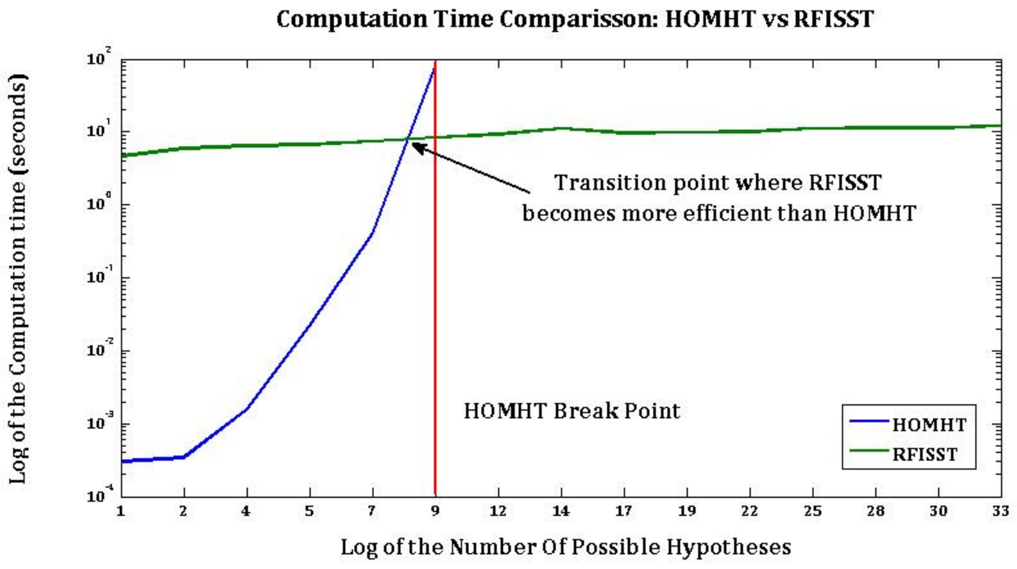


**Figure 6: Comparison of an MHT implementation against SSMCMC-based FISST (also known as RFISST)**

## 4. IMPLEMENTATION IN THE SCALA COMPUTING LANGUAGE AND ON THE MAUI HPC

In order to speed up processing, we implemented two methods of parallelization. First, for a given set of input parameters, we utilized parallel collections from the Scala standard library. Scala abstracts instruction level parallelism at a high level making this implementation rather straightforward. Whenever a serial collection is used, we replace it with parallel collection and Scala handles the rest. This kind of instruction level parallelism can only take advantage of the maximum number of cores available on a specific processor through the Java Virtual Machine (JVM).

Secondly, we wrote an Apache Spark wrapper around the Scala code to implement an embarrassingly parallel algorithm structure. This was accomplished by constructing Spark resilient distributed datasets (RDDs) consisting of each permutation of input parameters. We were then able to execute the embarrassingly parallel algorithm on the Maui High Performance Computing Center Riptide machine which is an IBM iDataPlex rated at 251.6 peak TFLOPS. Riptide has 756 compute nodes with Direct Water Cooling, and each diskless compute node has two 8-core processors (16 cores) with its own Red Hat Enterprise Linux OS, sharing 32 GBytes of memory. Through the MHPCC queuing system, we were typically able to request 16 nodes (or a total of 256 cores) for short periods of time (hours) or 1 node with 16 cores for longer periods of time (days).

## 5. SIMULATION RESULTS

In this section, we describe the simulation parameters and the first set of preliminary results from the HPC runs. The simulation was run with varying number of randomly generated RSOs near the geostationary orbit with the uniformly randomly generated orbital parameter ranges (except for semi-major axis) shown in Table 1. Each RSO was given an initial Gaussian uncertainty with a standard deviation of 5 km (isotropically in all three position directions) and 0.1 m/s in velocity (isotropically in all three velocity directions). Unperturbed two-body dynamics were used for the true RSO and filter propagations.

Six ground sensors with representative and realistic sensor properties and locations were used in the simulation. Half the sensors were radar sensors and the other half optical that operate only during the nighttime. A UKF implementation of the FISST Bayesian recursions (Eq. (1) and Eq. (2)) was processed, where each RSO in each underlying hypothesis was propagated and updated according to the UKF. Two performance metrics are used to assess performance. The first is the total CPU time to simulate a 6-hour period. The second performance metric is the total average true tracking error. This metric is defined to be the true tracking error (in position only) for each RSO, summed over all the RSOs, and finally divided by the total number of RSOs.

**Table 1: Orbit Parameter Nominal Values and Ranges**

| Parameter | Range |
|---|---|
| Semi-Major Axis, km | 42,164.137 - 42,164.137 |
| Eccentricity | 0-0.1 |
| Inclination, degrees | 0-10 |
| Argument of Perigee, degrees | 0-0.1 |
| Right Ascension of the Ascending Node, degrees | 0-10 |
| True Anomaly, degrees | 0-180 |

The parameters that are varied from one run to the next are:

- Number of RSOs $N$: We have simulated five cases: 200 RSOs, 400 RSOs, 600 RSOs, 800 RSOs and 1000 RSOs
- Number of hypotheses $M$ to return by the data association algorithm as discussed earlier in the paper was set to 1
- Maximum number of SSMCMC iterations $L$: This is the maximum number of iterations allowed within the SSMCMC recursion
- Maximum number of convergence iterations $C$: This the maximum number of iterations that establishes stationarity of the returned associations

The results for the UKF implementation are shown in Figure 7-Figure 16. The following are notable observations:

- As one would expect, as the size of the problem or the maximum number of iterations for convergence and termination in SSMCMC increases, CPU time increases
- As one would expect, as the maximum number of iterations for convergence and termination in SSMCMC increases, the average global tracking error decreases (modulo random effects)
- As the number of RSOs increases, the average global error grows

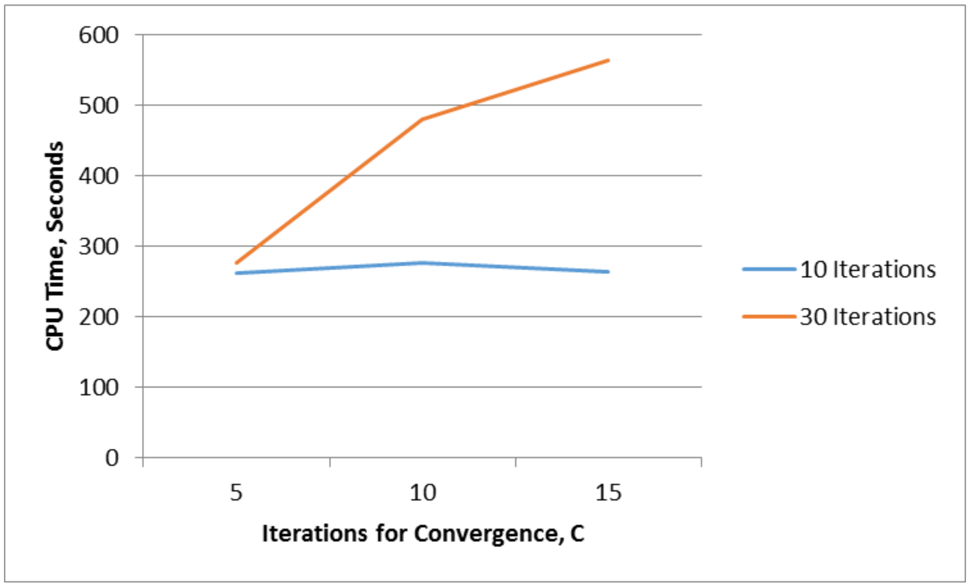


**Figure 7: CPU Time for a 200 RSO case, with $M$=1, and for $C = 5, 10, 15$ and $L = 10, 30$ with a UKF implementation**

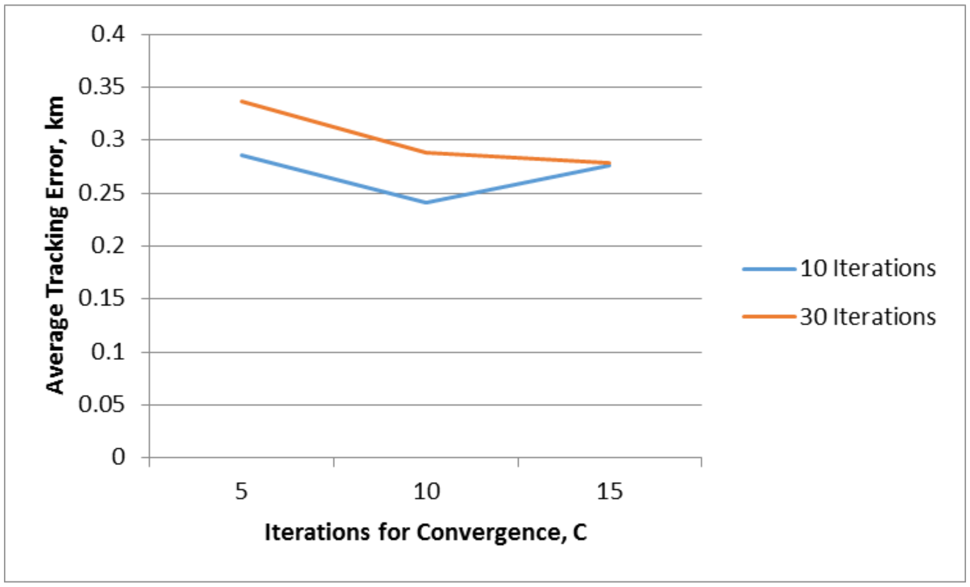


**Figure 8: Average tracking error for a 200 RSO case, with $M$=1, and for $C = 5, 10, 15$ and $L = 10, 30$ with a UKF implementation**

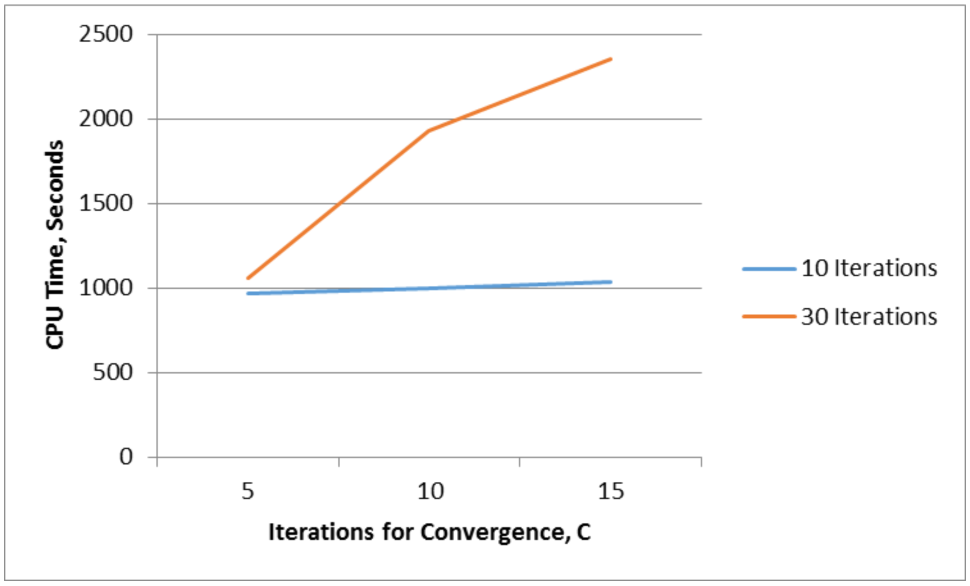


**Figure 9: CPU Time for a 400 RSO case, with $M$=1, and for $C = 5, 10, 15$ and $L = 10, 30$ with a UKF implementation**

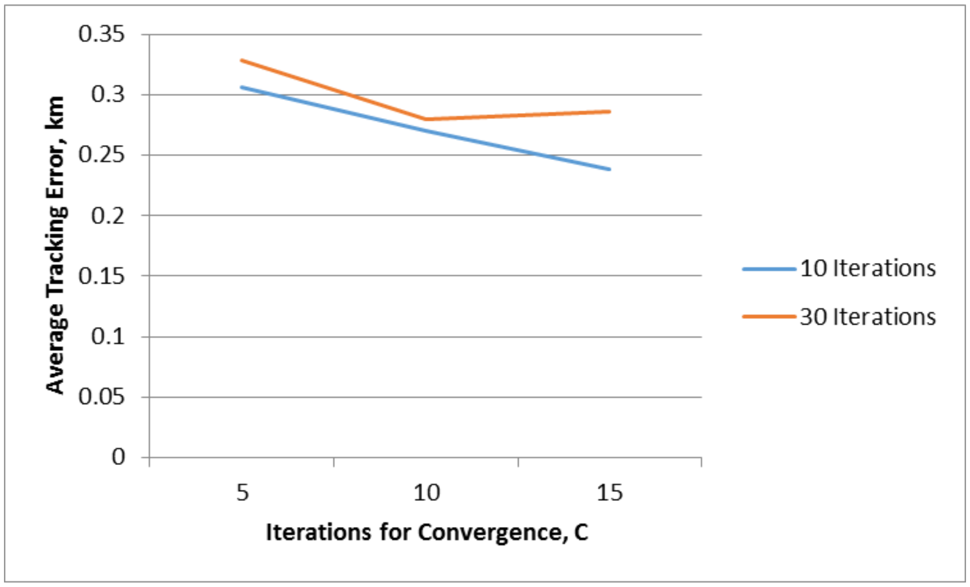


**Figure 10: Average tracking error for a 400 RSO case, with $M$=1, and for $C = 5, 10, 15$ and $L = 10, 30$ with a UKF implementation**

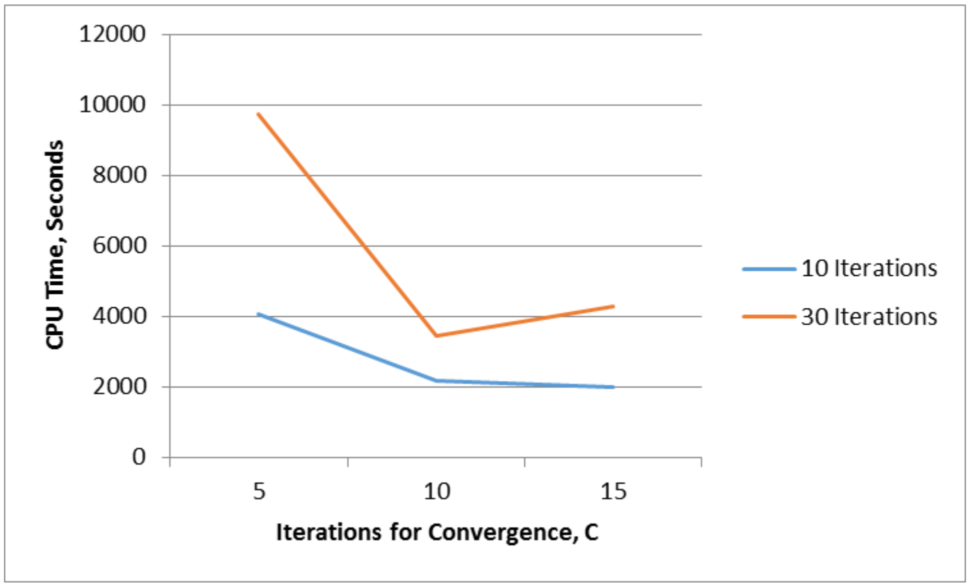


**Figure 11: CPU Time for a 600 RSO case, with $M$=1, and for $C = 5, 10, 15$ and $L = 10, 30$ with a UKF implementation**

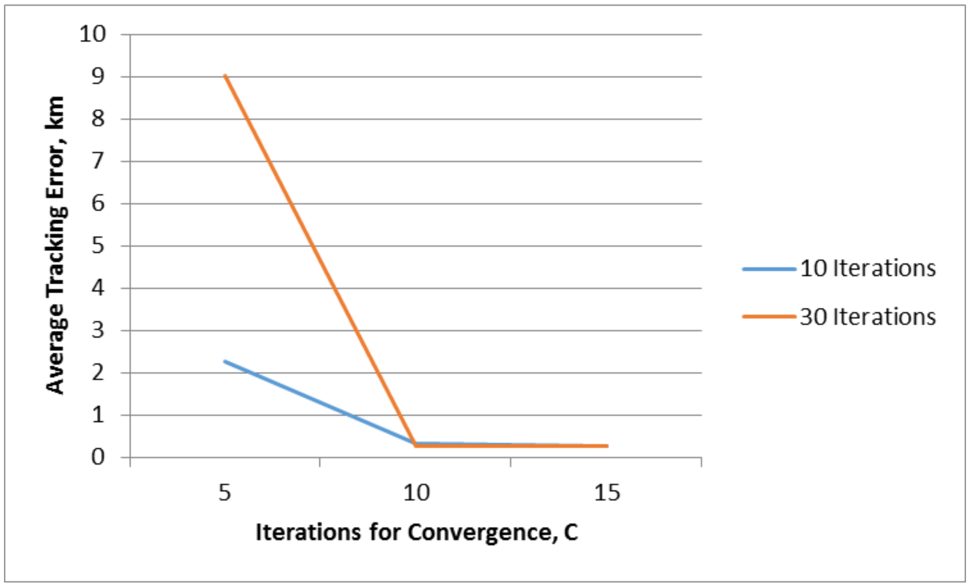


**Figure 12: Average tracking error for a 600 RSO case, with $M$=1, and for $C = 5, 10, 15$ and $L = 10, 30$ with a UKF implementation**

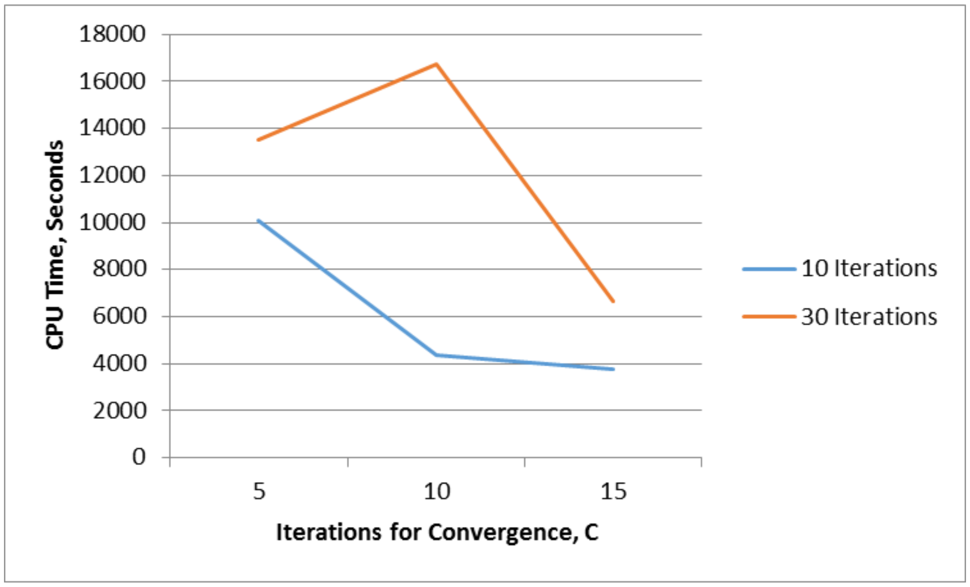


**Figure 13: CPU Time for a 800 RSO case, with $M$=1, and for $C = 5, 10, 15$ and $L = 10, 30$ with a UKF implementation**

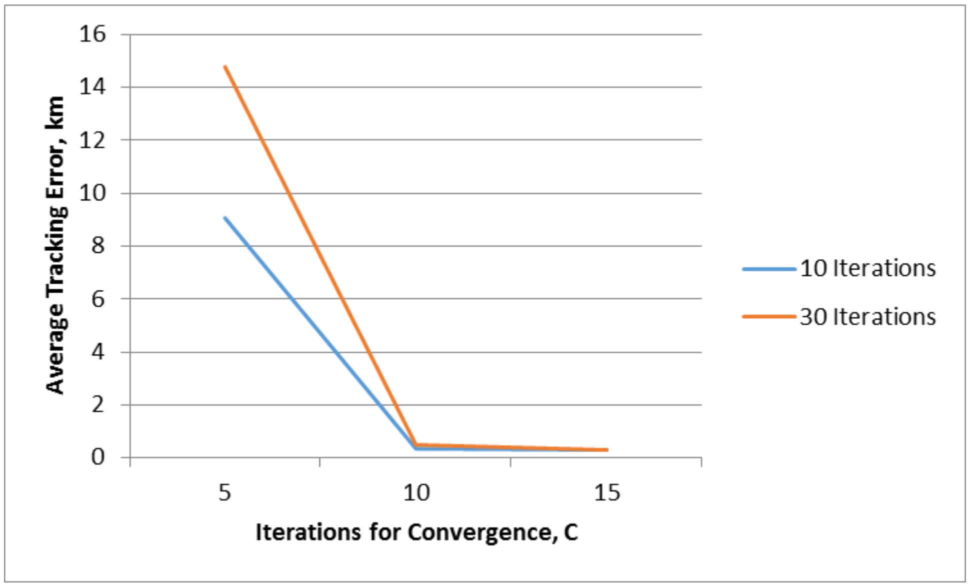


**Figure 14: Average tracking error for a 800 RSO case, with $M$=1, and for $C = 5, 10, 15$ and $L = 10, 30$ with a UKF implementation**

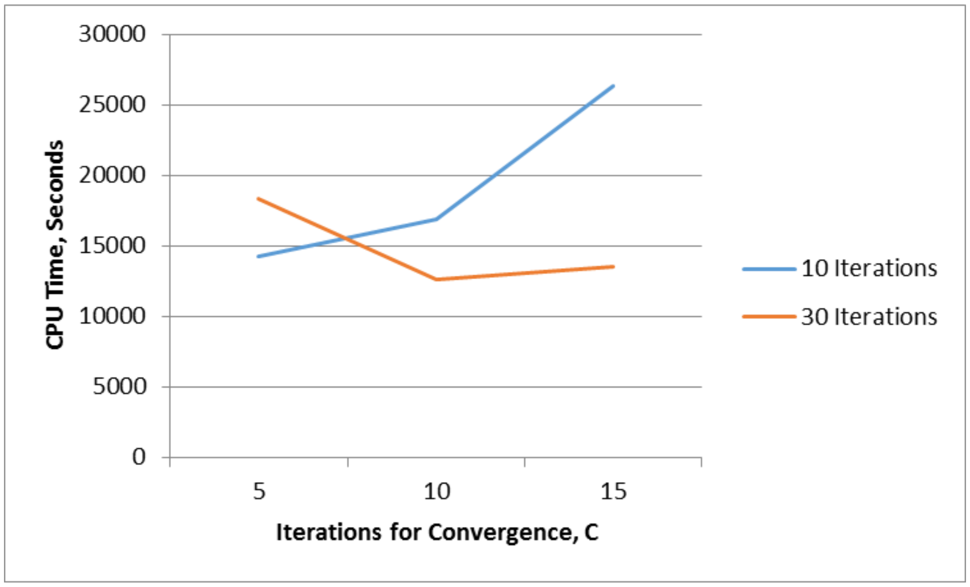


**Figure 15: CPU Time for a 1000 RSO case, with $M$=1, and for $C = 5, 10, 15$ and $L = 10, 30$ with a UKF implementation**

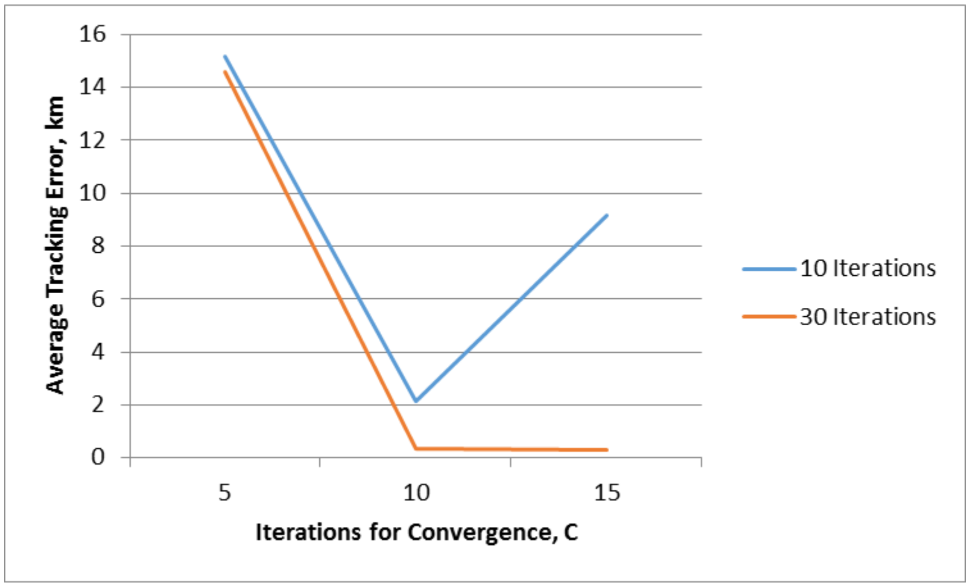


**Figure 16: Average tracking error for a 1000 RSO case, with $M$=1, and for $C = 5, 10, 15$ and $L = 10, 30$ with a UKF implementation**

## 6. CONCLUSION

In this paper we studied the performance of a scalable SSMCMC implementation of FISST on the Maui High Performance Computing facility. The analysis showed the effects of SSMCMC parameter selection and the size of the SSA problem (parameterized in terms of the number of RSOs in the study) on two key performance metrics. The first metric is computational time and the second was tracking error performance. As one would expect, the more hypotheses were retained by SSMCMC and the number of RSOs grew, computation time increased and tracking error, on average, dropped with the UKF implementation of SSMCMC-FISST. This is the first study implementing FISST on the HPC facility. A key outcome from this study is that SSMCMC-FISST is a viable approach to addressing the very large size of many of the key tasks underlying SSA. Future studies will include other types of hypotheses such as birth, death and object maneuver, as well as a careful performance comparison against alternative methods to multi-sensor, multi-object detection, tracking, identification and characterization in SSA.

## 7. REFERENCES

[1] J. Ferreira III, I. I. Hussein, J. Gerber and R. Sivilli, "Optimal SSN tasking to enhance real-time Space Situational Awareness," in *AMOS Conference*, Maui, HI, 2016.

[2] R. P. S. Mahler, Statistical Multisource-Multitarget Information Fusion, Artech House, 2007.

[3] I. R. Goodman, R. P. S. Mahler and H. T. Nguyen, Mathematics of Data Fusion, Kluwer Academic Publishers, 1997.

[4] Z. Sunberg, S. Chakravorty and R. Erwin, "Information space receding horizon control," *Trans. on Syst. Man Cybernet. B,* 2013.

[5] R. P. S. Mahler, Statistical Multisource-Multitarget Information Fusion, Artech House , 2007.

[6] H. G. Hoang, B. T. Vo and B. N. Vo, "A fast implementation of the generalized labeled multi-Bernoulli filter with joint prediction and update," in *Information Fusion,* Washington, D.C., 2015.

[7] W. Faber, S. Chakravorty and I. I. Hussein, "A Randomized Sampling based Approach to Multitarget Tracking," in *American Control Conference*, 2014.

[8] W. Faber, S. Chakravorty and I. Hussein, "A randomized sampling based approach to multi-object tracking," in *IEEE Conference on Information Fusion*, Washington DC, 2015.

[9] W. Faber, S. Chakravorty and I. Hussein, "Multi-object Tracking with Multiple Birth, Death, and Spawn Scenarios Using A Randomized Hypothesis Generation Technique (RFISST)," in *IEEE Information Fusion Conference*, Heidelberg, Germany, 2016.