High-throughput Binding Affinity Calculations at Extreme Scales

Jumana Dakka[=*], Matteo Turilli[=*], David W Wright[=†], Stefan J Zasada[=†],
Vivek Balasubramanian[*], Shunzhou Wan[†], Peter V Coveney[†] and Shantenu Jha[*,‡,§]
[*]Department of Electric and Computer Engineering, Rutgers University, NJ, USA
[†]Centre for Computational Sciences, UCL
[‡] Institute for Advanced Computational Sciences, Stony Brook University, NY, USA
[§]Computational Science Initiative, Brookhaven National Laboratory
[=] First authors, contributed equally, alphabetical order

***Abstract*—Resistance to chemotherapy and molecularly targeted therapies is a major factor in limiting the effectiveness of cancer treatment. In many cases, resistance can be linked to genetic changes in target proteins, either pre-existing or evolutionarily selected during treatment. Key to overcoming this challenge is an understanding of the molecular determinants of drug binding. Using multi-stage pipelines of molecular simulations we can gain insights into the binding free energy and the residence time of a ligand, which can inform both stratified and personal treatment regimes and drug development. To support the scalable, adaptive and automated calculation of the binding free energy on high-performance computing resources, we introduce the High- throughput Binding Affinity Calculator (HTBAC). HTBAC uses a building block approach in order to attain both workflow flexibility and performance. We demonstrate close to perfect weak scaling to hundreds of concurrent multi-stage binding affinity calculation pipelines. This permits a rapid time-to-solution that is essentially invariant of the calculation protocol, size of candidate ligands and number of ensemble simulations. As such, HTBAC advances the state of the art of binding affinity calculations and protocols.**

I. Introduction

In recent years, chemotherapy based on targeted kinase inhibitors (TKIs) has played an increasingly prominent role in the treatment of cancer. TKIs have been developed to selectively inhibit kinases involved in the signaling pathways that control growth and proliferation, which often become dysregulated in cancers. This targeting of specific cancers reduces the risk of damage to healthy cells and increases treatment success. Currently, 35 FDA-approved small molecule TKIs are in clinical use, and they represent a significant fraction of the $37 billion U.S. market for oncology drugs [**?**], [**?**]. Imatinib, the first of these of drugs, is partially credited for doubling survivorship rates in certain cancers [**?**], [**?**].

Unfortunately, the development of resistance to these drugs limits the amount of time that patients can derive benefits from their treatment. Resistance to therapeutics is responsible for more than 90% of deaths in patients with metastatic cancer [**?**]. While drug resistance can emerge via multiple mechanisms, small changes to the chemical composition of the therapeutic target (known as mutations) control treatment sensitivity and drive drug resistance in many patients (see Fig. **??**). In some commonly targeted kinases such as Abl, these changes account for as many as 90% of treatment failure [**?**].

There are two major strategies for countering the threat to treatment efficacy posed by resistance: tailoring the drug regimen received by a patient according to the mutations present in their particular cancer, and developing more advanced therapies that retain potency for known resistance mutations. In both cases, future developments require insight into the molecular changes produced by mutations, as well as ways to predict their impact on drug binding on a timescale much shorter than is typically experimentally feasible. This represents a grand challenge for computational approaches.

The rapidly decreasing cost of next-generation sequencing has led many cancer centers to begin deep sequencing of patient tumors to identify the genetic alterations driving individual cancers. The ultimate goal is to make individualized therapeutic decisions based upon these data—an approach termed *precision cancer therapy*. While several common (recurrent) mutations have been cataloged for their ability to induce resistance or for their susceptibility to particular kinase inhibitors, the vast majority of clinically observed mutations are rare. Essentially, this ensures that it will be impossible to make therapeutic decisions about the majority of individual patient tumors by using catalog-building alone.

Fortunately, concurrent improvements in computational power and algorithm design are enabling the use of molecular simulations to reliably quantify differences in binding strength. This provides the opportunity to use advances in molecular simulations to supplement existing inductive decision support systems with deductive predictive modeling and drug ranking [**?**], [**?**]. Where existing systems based on statistical inference are inherently limited in their range of applicability by the existence of data from previous similar cases, the addition of modeling allows evidence based decision making even in the absence of direct past experience.

The same molecular simulation technologies that can be employed to investigate the origins of drug resistance can also be used to design new therapeutics. Creating simulation protocols which have well defined uncertainty and produce statistically meaningful results represents a significant computational challenge. Furthermore, it is highly likely that differences among

investigated systems will demand different protocols as studies progress. For example, drug design programmes often require the rapid screening of thousands of candidate compounds to filter out the worst binders before using more sensitive methods to refine the structure. Not all changes induced in protein shape or behavior are local to the drug binding site and, in some cases, simulation protocols will need to adjust to account for complex interactions between drugs and their targets within individual studies.

Recent work that used molecular simulations to provide input to machine learning models [**?**] required simulations of 87 compounds even if they were designed merely to distinguish the highly active from weak inhibitors of the ERK2 kinase. If we wish to build on such studies to help inform later stages of the drug discovery pipeline, in which much more subtle alterations are involved, it is likely a much larger number of simulations will be required. This is before we begin to consider the influence of mutations or the selectivity of drugs to the more than 500 different genes in the human kinome [**?**].

For molecular simulations to make the necessary impact, the dual challenge of scale (thousands of concurrent multi-stage pipelines) and sophistication (adaptive selection of binding affinity protocols based upon statistical errors and uncertainty) will need to be tackled. Tools that facilitate the scalable and automated computation of varied binding free energy calculations on high-performance computing resources are necessary. To achieve that goal, we introduce the High-throughput Binding Affinity Calculator (HTBAC). HTBAC translates recent advances in workflow system building blocks to the application of rapid and accurate calculation of binding affinities. We demonstrate how HTBAC scales almost perfectly to hundreds of concurrent pipelines of binding affinity calculations on a leadership class machine. This permits the rapid time-to-solution that is essentially invariant of the size of candidate ligands as well as the type and number of protocols concurrently employed.

In the next Section, we provide details of ensemble molecular dynamics approach and its advantages over the single trajectory approach. We also introduce the ESMACS and related protocols to compute binding affinities using ensemble based approaches. In Section 3, we discuss the computational challenges associated with the scalable execution of multiple, and possibly concurrently executing protocols. Section 4 introduces RADICAL-Cybertools – a suite of building blocks to address the challenges outlined in Section 3. In Section 5 we introduce HTBAC and describe how it uses RADICAL-Cybertools to manage the execution of binding affinity calculations at extreme scales. Experiments to characterize the performance and scalability of HTBAC on the Blue Waters supercomputer are discussed in Section 6. We conclude with a discussion of the impact of HTBAC, implication for binding affinity calculations and near-term future work.

II. Methodology

The strength of drug binding is determined by a thermodynamic property known as the binding free energy (or binding affinity). One promising technology for estimating binding free energies and the influence of protein composition on them is molecular dynamics (MD) [**?**]. Our previous work [**?**], [**?**] has demonstrated that running multiple MD simulations based on the same system and varying only in initial velocities offers a highly efficient method of obtaining accurate and reproducible estimates of the binding affinity. We term this approach ensemble molecular dynamics, "ensemble" here referring to the set of individual (replica) simulations conducted for the same physical system. In this Section we discuss the advantages to this approach.

A. Ensemble Molecular Dynamics Simulations

Atomistically detailed models of the drug and target protein can be used as the starting point for MD simulations to study the influence of mutations on drug binding. The chemistry of the system is encoded in what is known as a potential [**?**]. In the parameterization of the models, each atom is assigned a mass and a charge, with the chemical bonds between them modeled as springs with varying stiffness. Using Newtonian mechanics the dynamics of the protein and drug can be followed and, using the principles of statistical mechanics, estimates of thermodynamic properties obtained from simulations of single particles.

The potentials used in the simulations are completely under the control of the user. This allows the user to manipulate the system in ways which would not be possible in experiments. A particularly powerful example of this are the so called "alchemical" simulations in which the potential used in the simulation changes, from representing a particular starting system into one describing a related target system during execution. This allows for the calculation of free energy differences between the two systems, such as those induced by a protein mutation.

MD simulations can reveal how interactions change as a result of mutations, and account for the molecular basis of drug efficacy. This understanding can form the basis for structure-based drug design as well as helping to target existing therapies based on protein composition. However, correctly capturing the system behavior poses at least two major challenges: The approximations made in the potential must be accurate enough representations of the real system chemistry; and sufficient sampling of phase space is also required.

In order for MD simulations to be used as part of clinical decision support systems, it is necessary that results can be obtained in a timely fashion. Typically, interventions are made on a timescale of a few days or, at most, a week. The necessity for rapid turn around times places additional demands on simulation protocols which need to be optimized to gain results with a short turn around time. Further to the scientific and practical considerations outlined above, it is vital that reliable uncertainty estimates are provided alongside all quantitative results for simulations to provide actionable predictions.

We have developed a number of free energy calculation protocols based on the use of ensemble molecular dynamics simulations with the aim of meeting these requirements [**?**], [**?**],

[?], [?]. Basing these computations on the direct calculation of ensemble averages facilitates the determination of statistically meaningful results along with complete control of errors. The use of the ensemble approaches however, necessitates the use of middleware to provide reliable coordination and distribution mechanisms with low performance overheads.

B. Protocols for Binding Affinity Calculations

We have demonstrated the lack of reproducibility of single trajectory approaches in both HIV-1 protease and MHC systems, with calculations for the same protein-ligand combination, with identical initial structure and force field, shown to produce binding affinities varying by up to 12 kcal mol $^{-1}$ for small ligands (flexible ligands can vary even more) [?], [?], [?]. Indeed, our work has revealed how completely unreliable single simulation based approaches are.

Our work using ensemble simulations have also reliably produced results in agreement with previously published experimental findings [?], [?], [?], [?], [?], [?], and correctly predicted the results of experimental studies performed by colleagues in collaboration [?]. While the accuracy of force fields could be a source of error, we know from our work to date that the very large fluctuations in trajectory-based calculations account for the lion's share of the variance (hence also uncertainty) of the results.

We designed two free energy calculation protocols with the demands of clinical decision support and drug design applications in mind: ESMACS (enhanced sampling of molecular dynamics with approximation of continuum solvent) [?] and TIES (thermodynamic integration with enhanced sampling) [?]. The former protocol is based on variants of the molecular mechanics Poisson-Boltzmann surface area (MMPBSA) end-point method, the latter on the 'alchemical' thermodynamic integration (TI) approach. In both cases, ensembles of MD simulations are employed to perform averaging and to obtain tight control of error bounds in our estimates. In addition, the ability to run replica simulations concurrently means that, as long as sufficient compute resources are available, turn around times can be significantly reduced compared to the generation of single long trajectories. The common philosophy behind the two protocols entails similar middleware requirements: In this work we focus on the ESMACS protocol but all results are applicable also to TIES.

Each replica within the ESMACS protocol consists of a series of simulation steps followed by post production analysis. Generally, an ESMACS replica will contain between 3 and 12 equilibration simulation steps followed by a production MD run, all of which are conducted in the NAMD package [?]. The first step is system minimization, the following steps involve the gradual release of positional constraints upon the structure and the heating to a physiologically realistic temperature. Upon completion of the MD simulation, free energy computation (via MMPBSA and potentially normal mode analysis) is performed using AmberTools [?], [?].

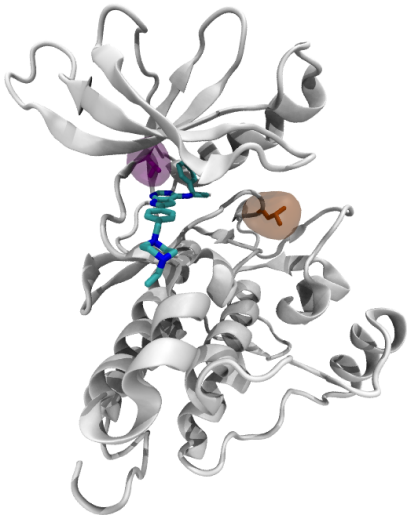

Fig. 1. Cartoon representation of the EGFR kinase bound to the inhibitor AEE788 shown in chemical representation (based on PDB:2J6M). Two residues implicated in modulating drug efficacy are highlights; in pink T790 and in orange L858. Mutations to either of these residues significantly alter the sensitivity to TKIs.

The ESMACS protocol is highly customizable. Both the number of simulation replicas in the ensemble and the lengths of their runs can be varied to obtain optimal performance for any given system. Using replicas that only vary in the initial velocities assigned to the atoms of the system we have defined a standard protocol which prescribes a 25 replica ensemble, each run consisting of 2 ns of equilibration and 4 ns of production simulation. Our protocol has produced bootstrap errors of below 1.5 kcal mol^{-1} (despite replica values varying by more than 10 kcal mol^{-1}) for a varied range of systems including small molecules bound to kinases and more flexible peptide ligands binding to MHC proteins [?], [?], [?]. In these systems, the error we obtained more than halves between ensembles of 5 and 25 replicas but increases in ensemble size have generally produced only small improvements. More generally though, there may be cases where it is important to increase the sampling of phase space either through expanding the ensemble or by considering multiple initial configurations.

The ESMACS protocol can also be extended to account for adaptation energies involved in altering the conformation of the protein or ligand during binding. Almost all MMPBSA studies in the literature use the so-called 1-trajectory method, in which the energies of protein-inhibitor complexes, receptor proteins and ligands are extracted from the MD trajectories of the complexes alone. The ESMACS protocol can additionally use separate ligand and receptor trajectories to account for adaptation energies, providing further motivation to deploy the protocol via flexible and scalable middleware.

C. Benchmark kinase system

A common target of kinase inhibitors is the epidermal growth factor receptor (EGFR) which regulates important cellular processes including proliferation, differentiation and apoptosis. EGFR is frequently over expressed in a range of

cancers, and is associated with disease progression and treatment. Clinical studies have shown that EGFR mutations confer tumor sensitivity to tyrosine kinase inhibitors in patients with non-small-cell lung cancer (examples shown in Fig. **??**) The tyrosine kinase domain of EGFR contains 288 residues, the full simulation system including solvent and the AEE788 inhibitor contains approximately 50 thousand atoms. The well established AMBER ff99SBildn and GAFF force fields [**?**], [**?**] were used to parameterize the system for this work.

D. Automated binding affinity calculations

The implementation of any physically realistic molecular simulation has always been an involved and multistage process, often requiring the scientist to overcome a large manual overhead in the construction, preparation, and execution protocols necessary to complete a set of simulations as well as to invoke various analysis protocols for determining desired properties post-production.

Several tools have been been developed to automate MD workflows for the rapid, accurate and reproducible computation of binding free energies of small molecules to their target proteins. For example, BAC [**?**] is a partially automated workflow system which comprises a) model building (including incorporation of mutations into patient specific protein models), b) running ensembles of MD simulations using a range of free energy techniques and c) statistical analysis. In Section 5, we decribed how we have enhanced BAC using the RADICAL-Cybertools to produce (HTBAC).

III. Computational Challenges at Extreme Scale

High-performance computing (HPC) environments were designed to primarily support the execution of single simulations. Current HPC platforms enable the strong and weak scaling of single tasks (hitherto mostly simulations), with limited software and systems support for the concurrent execution of multiple heterogeneous tasks as part of a single application (or workflow). As the nature of scientific inquiry and the applications to support that inquiry evolve, there is a critical need to support the scalable and concurrent execution of a large number of heterogeneous tasks.

Sets of tasks with dependencies that determine the order of their execution are usually referred to as "workflows". Often times, the structure of the task dependencies is simple and adheres to discernible patterns, even though the individual tasks and their duration are non-trivially distinct. Put together, it is a challenge to support the scalable execution of workflows on HPC resources due to the existing software ecosystem and runtime systems typically found.

Many workflow systems have emerged in response to the aforementioned problem. Each workflow system has its strengths and unique capability, however each system typically introduces its problems and challenges. In spite of the many successes of workflow systems, there is a perceived high barrier-to-entry, integration overhead and limited flexibility.

Interestingly, many commonly used workflow systems in high-performance and distributed computing emerged from an era when the software landscape supporting distributed computing was fragile, missing features and services. Not surprisingly, initial workflow systems had a monolithic design that included the end-to-end capabilities needed to execute workflows on heterogeneous and distributed cyberinfrastructures. Further, these workflow systems were typically designed by a set of specialists to support large "big science" projects such as those carried out at the LHC [**?**] or LIGO [**?**]. The fact that the same workflow would be used by thousands of scientists over many years justified, if not amortized, the large overhead of integrating application workflows with monolithic workflow systems. This influenced the design and implementation of interfaces and programming models.

However as the nature, number and usage of workflows has evolved so have the requirements: scale remains important but only when delivered with the ability to prototype quickly and flexibly. Furthermore, there are also new performance requirements that arise from the need to support concurrent execution of heterogeneous tasks. For example, when executing multiple homogeneous pipelines of heterogeneous tasks, for reasons of efficient resource utilization there is a need to ensure that the individual pipelines have similar execution times. The pipeline-to-pipeline fluctuation must be minimal while also managing the task-to-task runtime fluctuation across concurrently executing pipelines.

Thus the flexible execution of heterogeneous ensembles of MD simulations face both system software and middleware challenges: existing system software that is typically designed to support the execution of single large simulations on the one hand, and workflow systems that are designed to support specific use cases or 'locked-in' end-to-end executions. In the next section, we discuss the design and implementation of the RADICAL-Cybertools, a set of software building blocks that can be easily composed to design, implement and execute domain specific workflows rapidly and at scale.

IV. RADICAL-Cybertools

We have designed RADICAL-Cybertools (RCT) to address some of the challenges in developing and executing workflows on HPC platforms. RCT consists of three main components: Ensemble Toolkit (EnTK) [**?**], RADICAL-Pilot (RP) [**?**] and RADICAL-SAGA (RS) [**?**]. EnTK provides the ability to create and execute ensemble- based workflows/applications with complex coordination and communication but without the need for explicit resource management. EnTK uses RP which provides resource management and task execution capabilities, and RP uses RS as an interoperable resource access interface.

RCT eschew the concept of a monolithic workflow systems and uses "building blocks". RCT provide scalable implementations of building blocks in Python that are used to support dozens of scientific applications on high-performance and distributed systems [**?**], [**?**], [**?**], [**?**], [**?**]. In this section we discuss details of RP and EnTK, and understand how these components have been used to support the flexible and scalable execution of pipelines, which will permit a deeper understanding of HTBAC in the next section.

A. RADICAL-Pilot

The scalable execution of applications with large ensembles of tasks is challenging. Traditionally, two methods are used to execute multiple tasks on a resource: each task is scheduled as an individual job, or MPI capabilities are used to execute multiple tasks as part of a single job. The former method suffers from unpredictable queue time: each task independently awaits in the resource's queue to be scheduled. The latter method relies on the fault tolerance of MPI, and is suitable to execute tasks that are homogeneous and have no interdependencies.

The Pilot abstraction [**?**] solves these issues: The pilot abstraction: (i) uses a placeholder job without any tasks assigned to it, so as to acquire resources via the local resource management system (LRMS); and, (ii) decouples the initial resource acquisition from task-to-resource assignment. Once the pilot (container-job) is scheduled via the LRMS, it can pull computational tasks for execution. This functionality allows all the computational tasks to be executed directly on the resources, without being queued via the LRMS. The pilot abstraction thus supports the requirements of task-level parallelism and high-throughput as needed by science drivers, without affecting or circumventing the queue policies of HPC resources.

RADICAL-Pilot is an implementation of the pilot abstraction, engineered to support scalable and efficient launching of heterogeneous tasks across different platforms.

B. Ensemble Toolkit

An ensemble-based application is a **workflow**, i.e., a set of tasks with dependencies that determine the order of their execution. Subsets of these tasks can be **workloads**, i.e., tasks whose dependencies have been satisfied at a particular time and may be executed concurrently. Ensemble-based application vary in the type of coupling between tasks, the frequency and volume of information exchanged between these tasks, and the executable of each task. This type of applications requires specific coordination, orchestration and execution protocols, posing both domain-specific and engineering challenges.

Ensemble Toolkit (EnTK), the topmost layer of RCT, simplifies the process of creating and executing ensemble-based applications with complex coordination and communication requirements. EnTK decouples the description of ensemble-based applications from their execution by separating three orders of concern: specification of task and resource requirements; resource selection and acquisition; and task execution management. Domain scientists retain full control of the implementation of their algorithms by programming ensemble-based applications by describing what should be executed, when it should be executed and where it should be executed. EnTK automates the process of acquiring the resources needed by those applications and managing the task execution via a runtime system like RADICAL-Pilot.

EnTK has been previously used to develop the ExTASY [**?**] framework for the execution of MD-based advanced sampling methods. EnTK is also being utilized to develop ensemble-based applications to execute workflows in the seismology and climate science domains. Each application requires different workflows, with different types of tasks, executed on specific HPC resources. EnTK supported the development of a simulation-analysis execution pattern for ExTASY, implementing two sampling algorithms for execution of both multithreaded and MPI tasks on ARCHER, Stampede and Blue Waters HPC machines.

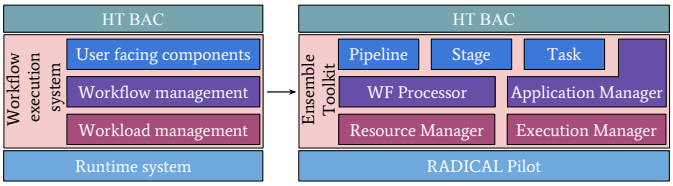

Fig. 2. **Left:** Ensemble Toolkit overview, showing how the abstract workflow execution system is mapped to specific components exposed to the users and components internal to the toolkit.

Currently, EnTK enables the execution of up to $O(10^4)$ tasks on thousands of CPU and GPU nodes on the resources of XSEDE, DOE leadership machines at ORNL and Blue Waters at NCSA. At this scale, multiple sources of failure affect execution, producing loss of information and compute time. Thus, fault tolerance and recovery capabilities are first order concerns of the design of EnTK. As such, EnTK can handle task execution failures, resource-level failures, and failures of its own subcomponents and subprocesses, including network connectivity issues.

EnTK exposes to the end-user four components, namely **Resource Handle**, **Pipeline**, **Stage** and **Task**, to create ensemble-based applications. Users also use the **Application Manager** component to specify resources for the execution of their ensemble-based application. Two internal components called **WF Processor** and **Execution Manager** are used to coordinate and enact the execution of the application (see Figure **??**).

The Task component is used to encapsulate an executable and its input/output data, resource and software environment requirements. The Stage component is used to group tasks that have the same dependencies and can be executed concurrently. The Pipeline component is used to describe a sequence of stages, i.e., operations that need to be executed sequentially, not concurrently.

EnTK enables users to program ensemble-based applications directly in Python, without requiring knowledge of a domain-specific language. The use of the Task, Stage, and Pipeline components, which are based on the well known set and list data structures, avoids the need to express explicitly relationship among tasks. These relationships are insured by design, depending on the partitioning of the set of tasks into stages and then pipelines. Further, EnTK enables an explicit definition of pre and post conditions on the execution of tasks, stages, and pipelines, enabling a fine grained adaptivity, both a local and global level. Conveniently, this does not require the codification of a directed acyclic graph (DAG), a process that imposes a rigid representation model on the domain scientists.

Once the workflow of an ensemble-based application is de-

scribed, one or more pipelines are submitted to the Application Manager module for execution. The Application Manager sets up multiple processes, threads and a messaging infrastructure for communication. The WF Processor processes each pipeline, identifying tasks which have their dependencies satisfied and can be executed concurrently. These tasks are handed over to the Execution Manager that binds tasks to resources on the basis of a set of heuristics called "execution strategies". The Execution Manager uses the underlying runtime system, RADICAL-Pilot, to execute the tasks on the specific target resource.

EnTK is implemented as a multiprocess application. The number of processes can be tuned to match the throughput requirements of ensemble-based applications and the capabilities of diverse target resources. All but the master process of the toolkit are stateless, enabling process-level fault tolerance. Upon failure of one or more stateless processes, application tasks can be assigned to existing or newly created processes without blocking the overall execution. Upon failure of the master process, execution can be restarted without loosing data about tasks that have been already executed.

V. High Throughput Binding Affinity Calculator

HTBAC uses RCT to implement ESMACS, and similar protocols such as TIES. These protocols consist of pipelines with stages comprised of heterogeneous tasks. For example, equilibration and production, followed by post processing steps. The different protocols differ in the details of the pipelines, stages and synchronization [**?**] they are made up of.

RADICAL-Cybertools provides advanced resource management capabilities and, thereby delivers the necessary high-throughput capabilities required.HTBAC is integrated with the EnTK component of RCT. EnTK provides a common API, execution and programming model, thus allowing HTBAC to express the workflows associated with different protocols uniformly, and thus minimize development effort and complexity.

The Ensemble Toolkit API exposes four components (**Resource Handle**, **Pipeline**, **Stage**, and **Task**) to express the application logic of HTBAC. We describe these components for the ESMACS protocol. The concept of an ensemble in the ESMACS protocol maps directly to a set of pipelines in EnTK, where each pipeline contains functions that operate on a given replica. EnTK interprets these replicas as independent pipelines. Each pipeline consists of multiple stages representing a well-defined execution order; each stage can contain heterogeneous workloads. Although each stage of a pipeline depends on its predecessor, the pipelines execute independently of each other. The patterns within pipelines are identical and describe an ensemble of replica simulations, as shown in Fig **??**.

Each stage is composed of a single unique task which is described by a set of attributes that define the workload parameters such as the location of input files, the number of simulations and the MD engine(s) to launch simulations. The ESMACS protocol defines 7 stages, in which the first and last stages perform staging of the input/output data. The middle

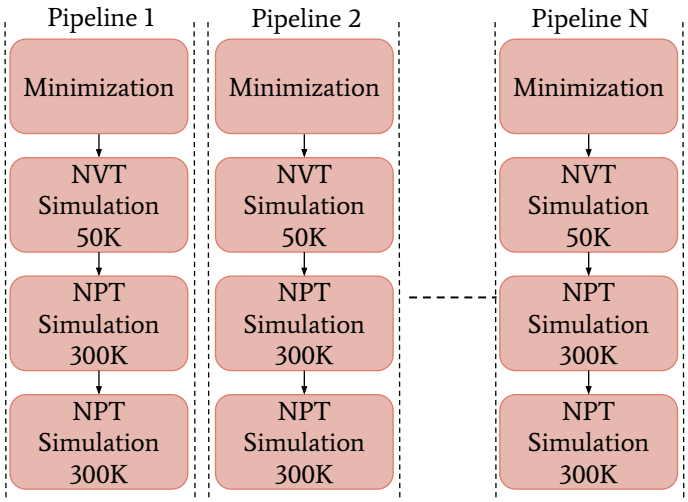

Fig. 3. ESMACS protocol indicating how an N replica ensemble is implemented in HTBAC. Each replica is mapped to a single EnTK pipeline. Each pipeline is equivalent and represents a set of simulations which are captured as stages by EnTK.

stages indicate simulation tasks as shown in Fig **??**. The task is appended to a stage and stages are appended to a pipeline to maintain temporal order. The workflow relies on a resource configuration which consists of the details required to use a resource where the application will be executed including runtime, queue, and account details. We capture the integration of the application (ESMACS protocol) and how it interfaces with EnTK in Fig **??**.

We define the client resource in Fig **??** as the workload system—HTBAC which describes a series of replicas with ordered functions as a pipelines with stages and tasks. EnTK interprets these pipelines as a functional set of tasks and generates the pilot description that contains the resource configuration of how to run the HTBAC workload. For the ESMACS protocol running on Blue Waters we define the runtime system, queue, and the pilot size. Once RADICAL-Pilot receives this new workload it generates a pilot that submits placeholders to the queue. Once the pilot is activated, the RP-Agent submits the tasks in the form of compute units to the placeholders to begin execution.

VI. HTBAC Experiments

Before embarking on a computational campaign that will consume 150M core hours on the NCSA Blue Waters machine, we studied the scalability of HTBAC so as to determine optimal workflow sizing and resource utilization for the ESMACS protocol. The goal is twofold: (1) understanding the invariance of HTBAC execution time over the number of workflow pipelines executed; and (2) studying how the performance of EnTK and RP varies in relation to the size of workflow.

A. Experiment Design

We designed two experiments to measure HTBAC, EnTK and RP weak scalability when executing an increasing number of concurrent pipelines. According to the use case described in

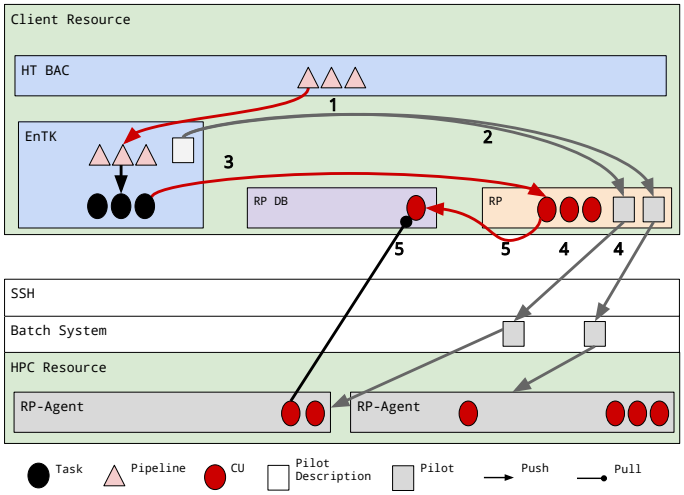

Fig. 4. **Integration between HTBAC workflow and EnTK. Numbers indicate the temporal sequence of execution. RADICAL-Pilot (RP) database (DB) can be deployed on any host reachable from the resources.**

Section **??**, each pipeline consists of seven stages, each stage with a single task. EnTK manages the queuing of the tasks in accordance with the order and concurrency mandated by stages and pipelines: For each pipeline, each stage is executed sequentially while pipelines are executed concurrently.

Experiment 1 measures the baseline behavior of EnTK and RP with the workflow of the ESMACS protocol and a null workload (/bin/sleep 0). The goal is to isolate the overheads of EnTK and RP from the specifics of the executables of the workflow and the overheads of the resources. The null workload does not require data staging, I/O on both memory and disk, or communication over network.

Experiment 2 replicates the design of Experiment 1 but it executes the workflow of the ESMACS protocol with the actual simulation and data for the EGFR kinase workload. The comparison between the two experiments enables performance analysis of EnTK and RP to understand whether and how the size of the executed workflow affects its overheads. Further, Experiment 2 shows also whether HTBAC execution time is sensitive to the number of concurrent pipelines executed.

Both experiments measure the weak scalability of HTBAC, EnTK and RP. This means that the ratio between the number of pipelines and cores is kept constant by design. While an investigation of strong scalability would contribute to a better understanding of the behavior of HTBAC, EnTK and RP, it is of limited interest for the current use case. The driving goal of HTBAC is to increase throughput by a means of concurrency of pipelines, not in the number of sequential executions per core. This is a driving motivation to target large HPC machines instead of so-called HTC infrastructures.

B. Experiment Setup

We perform both Experiment 1 and 2 on NCSA's Blue Waters—a 13.3 petaFLOPS Cray, with 32 Interlago cores/50 GB RAM per node, Cray Gemini, Lustre shared file system. Currently, we exclusively use CPUs on Blue Waters as GPUs are not required by our use case. RCT support the use of both type of architectures and we previously benchmarked the use of GPUs.

We perform our experiments from a virtual machine hosted in Europe. This helps to simulate the conditions in which the experimental campaign will be performed by the research group at UCL. This is relevant because, as most HPC resources, Blue Waters does not allow for executing applications on the login node of the cluster. To this end, RCTs support gsissh for X509 authentication and authorization.

Table **??** shows the setup for Experiment 1 and 2. The ESMACS protocol is executed with up to 25 concurrent but independent pipelines and therefore their concurrent execution does not entail communication overhead. Further, the EGFR kinase studies can benefit from greater concurrency because potential HTBAC users may wish to extend their protocols beyond the current scale of ESMACS. Consistently, our experiments push the boundaries of current scale by executing 8, 16, 32, 64 and 128 concurrent pipelines.

EnTK uses RP to acquire resources via a single pilot. The size of the pilot is contingent upon characterization of performance, in this case, weak scalability. Accordingly, we request the maximum number of cores required by the workload as the number of cores in a pilot. We use between 64 and 1024 cores in Experiment 2 as the NAMD executable used in stages 3, 4, 5, and 6 requires 8 cores. Stages 1, 2 and 7 require instead 1 core. The null workload of Experiment 1 requires only 1 core per stage but we request the same number of cores as for Experiment 2 to be able to compare the overheads of both EnTK and RP across experiments.

All experiments use EnTK version 0.4.7 and RP version 0.42. The MD engine used is NAMD-MPI. The equilibration tasks of stage 4 and 6 are assigned 5000 timesteps while the task of stage 5 requires 55000 timesteps. We ran two trials of both the null and MD workload at each pipeline configurations.

C. Results

First we characterize the overhead of EnTK and RP in the null workload, where we isolate the overhead introduced by the two systems (Figure **??**). We see a (slightly) superliner increase of EnTK overhead, between 0.1 and 1.8 seconds. This overhead depends on the number of tasks that need to be translated in-memory from a Python object to a compute unit (CU) description. As such, it is expected to grow proportionally to the number of tasks, barring some competition of resources.

RP overhead is also, on average, superlinear but with a much greater variance. This variance is due to mainly two factors: Network latency and filesystem latency on the HPC resource. EnTK submits CU descriptions to the MongoDB used by RP, and the RP pilot pulls these descriptions from the same database. As described in Section **??**, this pull operation occurs over a wide area network, introducing varying amounts of latency. Further, RP writes and reads the CU descriptions multiple times to and from the shared filesystem of the HPC

TABLE I

EXPERIMENT 1 EXECUTES THE 7 STAGES OF THE ESMACS PROTOCOL WITH A NULL WORKLOAD; EXPERIMENT 2 USES THE ACTUAL MD WORKLOAD OF THE ESMACS PROTOCOL. ESMACS PROTOCOL WITH EGFR KINASE WORKLOAD: (1) UNTAR CONFIGURATION FILES; (2) PREPREP; (3) MINIMIZE WITH DECREASING RESTRAINTS; (4) EQUILIBRATION: NVT SIMULATION AT 50K, WITH RESTRAINTS; (5) EQUILIBRATION: NPT SIMULATION AT 300K, WITH DECREASING RESTRAINTS; (6) EQUILIBRATION: NPT AT 300K, NO CONSTRAINTS; (7) TARBALL OUTPUT FILES.

Experiment ID	Protocol	Workload	# Trials	# Pipelines	# Stages	# Tasks	# Cores per pilot
1	ESMACS	Null workload	2	8, 16, 32, 64, 128	7	7	64, 128, 256, 512, 1024
2	ESMACS	EGFR kinase Workload	2	8, 16, 32, 64, 128	7	7	64, 128, 256, 512, 1024

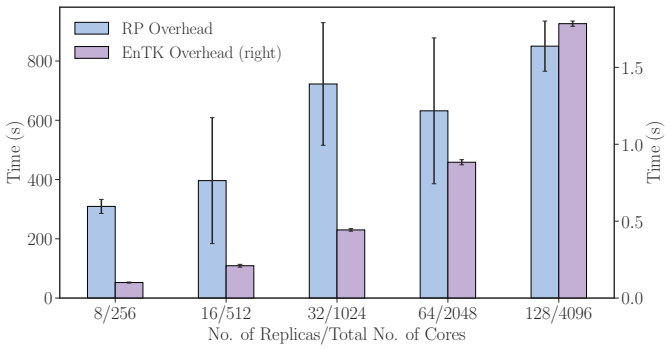

Fig. 5. Overheads of Ensemble Toolkit (EnTK) and RADICAL-Pilot (RP) when executing HTBAC using a null workload. We plot the baseline EnTK/RP overheads without the application workload across two trials per pipeline configuration.

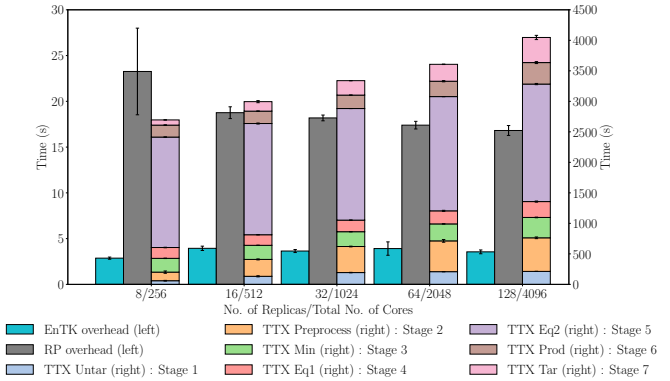

Fig. 6. Introducing the use-case EGFR kinase workload, we observe similar EnTK/RP overhead behavior as with the null workload with higher values as the number of pipelines increases. We show a breakdown of TTX of each stage (Stage 1 - Stage 7). Across pipeline configurations, TTX and RP overheads (accounting for the error bars) show weak scaling performance.

machine. Together, these two factors introduce delays in the scheduling of the CUs.

When the workload includes the EGFR kinase, we see (Figure **??**) that the RP overhead becomes on average less than 10% of the average total execution time (TTX), defined as $TTX = TTC - T_q$ where TTC is time-to-completion and T_q is time spent queuing on the HPC machine. We further break down TTX into the time-to-completion per stage, where stages 1,2, and 7 perform file movements, while stages 3,4,5, and 6 execute NAMD tasks. At this level we notice that the time-to-completion of the NAMD stages are essentially invariant across pipelines of different size while file movement stages exhibit linearly increasing behavior. In addition, when accounting for variance, RP overheads also show linear weak scaling behavior. As expected, EnTK overhead remains super-linear and comparable to the one measured in Experiment 1. This is because in both experiments EnTK overhead depends on the number of tasks translated to CU descriptions.

Experiments 1 and 2 show how the overheads of both EnTK and RP tend to be invariant across type of workload executed. Their scaling behavior and, to some approximation, their absolute values are comparable between Figure **??** and **??**. This is relevant because it shows that the systems used to coordinate and execute the ESMACS protocol add a constant and comparatively not relevant overhead to the execution of NAMD.

VII. DISCUSSION AND CONCLUSION

It is necessary to move beyond the prevailing paradigm of running individual MD simulations, which provide irreproducible results and cannot provide meaningful error bars [**?**]. Further, the ability to flexibly scale and adapt ensemble-based protocols to the systems of interest is vital to produce reliable and accurate results on timescales which make it viable to influence real world decision making. To meet these goals, we have designed and developed the high-throughput binding affinity calculator (HTBAC).

HTBAC employs the RADICAL-Cybertools building blocks for workflow systems. We demonstrate how HTBAC scales almost perfectly to hundreds of concurrent pipelines of binding affinity calculations on a leadership class machine. This permits the rapid time-to-solution that is essentially invariant of the size of candidate ligands as well as the type and number of protocols concurrently employed. The use of software abstractions ("building blocks") future proofs users of HTBAC to evolving hardware platforms, while providing immediate benefits of scale and support for a range of workflows. Thus, HTBAC represents an important advance towards the use of molecular dynamics based free energy calculations to the point where they can produce actionable results both in the clinic and industrial drug discovery.

In the short term, the development of HTBAC will allow a significant increase in the size of study. Much of the literature on MD based free energy calculations is limited to a few tens of systems, usually of similar drugs bound to the same protein target. By facilitating investigations of much larger datasets, HTBAC also provides a step towards tackling grand challenges in drug design and precision medicine, where it is necessary to understand the influences on binding strength for hundreds

or thousands of drug-protein variant combinations. Only in aiming to meet this ambitious goals we will be able to reveal the limits of existing simulation technology and the potentials used to approximate the chemistry of the real systems.

Software and Data Ensemble toolkit can be found at http://radicalensemblemd.readthedocs.io/en/latest/ and is released under the MIT license. Raw data and scripts to reproduce experiments can be found at https://github.com/radical-experiments/htbac-experiments.

ACKNOWLEDGEMENTS

Access to Blue Waters was made possible by NSF OAC 1713749. The software capabilities were supported by RADICAL-Cybertools (OAC 1440677) and NSF ICER 1639694. We thank Andre Merzky and other members of the RADICAL team for support. PVC, SJZ and DWW would like to acknowledge the support of the EU H2020 CompBioMed (675451), EUDAT2020 (654065) and ComPat (671564) projects.

Author Roles and Contribution: PVC and SJ conceived this work. MT and SJ designed the experiments with support from JD and VB. JD performed the bulk of experiments on Blue Waters, with support from DWW and VB. VB developed the version of EnTK used for experiments. MT, VB, JD and SJ analyzed the data. SJZ initially developed the ESMACS protocol using EnTK. DWW, MT and SJ wrote the bulk of the paper.