High-throughput Binding Affinity Calculations at Extreme Scales

Author 1*, Author 2†, Author 3‡, Author 4§,
Author 5*, Author 6†, Author 7‡ and Author 8§
*Affiliation 1
†Affiliation 2
‡Affiliation 3
§Affiliation 4

Abstract*—Resistance to chemotherapy and molecularly targeted therapies is a major factor in limiting the effectiveness of cancer therapies. 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 the understanding of the molecular determinants of drug binding. The key physical parameters determining drug effectiveness are the binding free energy and the residence time of a ligand. Using molecular simulation we can gain insights into both of these quantities, which can inform both stratified or personal treatment regimes and drug development. The level of sampling required to obtain accurate and reproducible binding affinities and kinetic information can only efficiently be obtained in a timely fashion through the use of ensembles of many simulations executed on HPC resources. In state of the art calculation techniques each simulation may, in fact, consist of a complex multi-stage workflow. The resulting trajectories reveal in atomic detail how interactions and protein behaviour change as a result of mutations, and account for the molecular basis of drug efficacy. There is a need to study a wide range of cancer drugs and candidate ligands in order to support personalised clinical decision making based on genome sequencing and drug discovery, using automated workflows underpinned by ensemble-based high performance computing methods running at unprecedented scales. To support this scalable and automated execution of workflows on leadership class resources, this paper introduces the High- Throughput Binding Affinity Calculator (HTBAC). HTBAC allows the investigation .... HTBAC provides the ability to separate the "setting-up" (i.e., defining) of a workflow from its "execution".

I. Introduction

Cancer is recognized as a major public health problem throughout the world and is the second most common cause of death in the United States. [?] In recent years chemotherapy based on targetted kinase inhibitors (TKIs) has played an increasingly prominent role in the treatment of cancer. The design of TKIs was inspired by modern genetic understanding of DNA, the cell cycle, and molecular signaling pathways and they have been developed to selectively inhibit kinases involved in signaling pathways controling growth and proliferation which often become dysregulated in cancers. This targetting 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 for the past decade, they have represented 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 Figure 1). 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 this threat to treatment efficacy: tailoring the drug regimen received by a patient according to the mutations present in their particular cancer, and development of 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 patient tumors to identify the genetic alterations driving individual cancers, with the ultimate goal of making individualized therapeutic decisions based upon this data – an approach termed *precision cancer therapy*. While several common (recurrent) mutations have been catalogued due to their ability to induce resistance or susceptibility to particular kinase inhibitors, the vast majority of clinically observed mutations are rare, essentially ensuring that it will be impossible that catalog-building alone will be sufficient for making therapeutic decisions about the majority of individual patient tumors. Fortunately, concurrent improvements in computational power and algorithm design mean that reliably quantifying differences in binding strength from molecular simulation is now becoming a genuine possibility. This provides the opportunity to use automated simulation workflows to supplement existing inductive ('Baconian') decision support systems with deductive ('Popperian') predictive modelling and drug ranking [?], [?]. Where

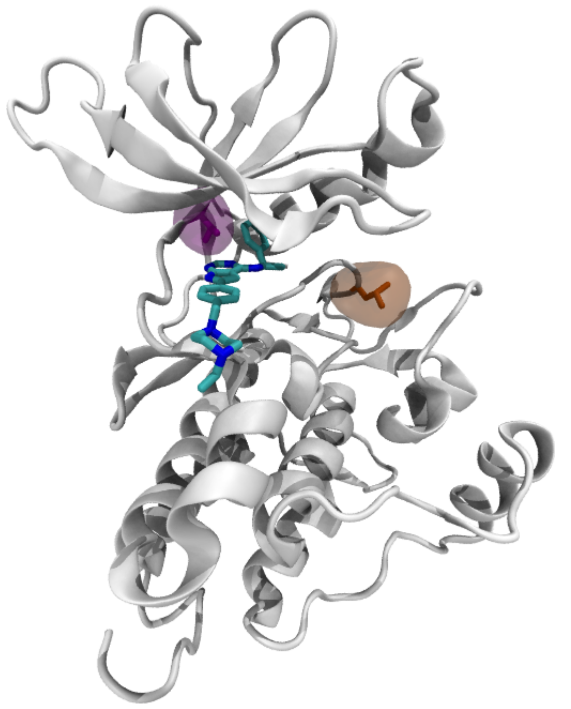

Fig. 1. Cartoon representation of the EGFR kinase bound to the inhibitor AEE788 shown in chemical representation (based on PDB:2J6M). Two residues implicated in modulating drug efficacy are highlights; in pink T790 and in orange L858. Mutations to either of these residues significantly alter the sensitivity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

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 modelling 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 meet the varying practical demands of different applications (such as the rapid turn around times needed for clinical decision support) whilst obtaining statistically meaningful results represents a significant computational challenge. Furthermore, it is highly likely that differences between the varied systems which might be investigated will demand different strategies can be employed as studies progress. For example in drug design programmes it is usual to need to rapidly screen thousands of candidate compounds to filter out the worst binders before more sensitive methods are required when refining the structure. Not all changes induced in proteins shape or behaviour 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 [**?**] designed merely to distinguish the highly active from weak inhibitors of the ERK2 kinase required simulations of 87 compounds. 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 would be required. This is before we begin to consider the influence of mutations or the selectivity of drugs to the more than five hundred different genes in the human kinome. [**?**] In order to tackle these grand challenges using molecular simulation is likely to require studies executing thousands of compute intensive runs. In order to contemplate such a programme of work it is necessary to have computational tools that facilitate the execution and coordination of varied workloads without incurring serious performance overheads.

II. Methodology

A. Ensemble Molecular Dynamics Simulations

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). In order to study the influence of mutations on drug binding atomistically detailed models of the drug and target protein can be used as the starting point for MD simulations. 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 modelled 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 which allows the user to manipulate the system in ways which would not be possible in experiments. A particularly powerful example of this is so called “alchemical” simulations in which the potential used in the simulation changes from that representing one system to another during execution — allowing 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. Major challenges however exist in correctly capturing the system behaviour. Not only is it necessary that the approximations made in the potential are accurate enough representations of the real system chemistry but 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 optimised to gain results within a short turn around time.

Further to the scientific and practical considerations outlined above, in order for simulations to provide actionable predictions it is vital that robust and reliable uncertainty estimates are provided alongside all quantitative results. 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 consequence of the use of the ensemble approach is the need to automate the execution, analysis and data transfer of a large number of simulations. This computational pattern necessitates the use of middleware that provides 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. Whilst our work using ensemble simulations have reliably produced results in agreement with previously published experimental findings [?], [?], [?], [?], [?], [?], and correctly predicted the results of experimental studies performed by colleagues in collaboration [?]. While 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.

ESMACS: We have 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 is based on variants of the molecular mechanics Poisson-Boltzmann surface area (MMPBSA) end-point method and the latter the 'alchemical' thermodynamic integration (TI) approach. In both cases ensembles of MD simulations are employed in order 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 single long trajectories. The common philosophy behind the two protocols lead to similar middleware requirements, consequently in this work we focus on the ESMACS protocol but all results are applicable to both.

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 [?], [?], [?].

The ESMACS protocol is highly customizable and both the number of replica simulations 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. For a varied range of systems including small molecules bound to kinases and more flexible peptide ligands binding to MHC proteins this has produced bootstrap errors of below 1.5 kcal mol^{-1} (despite replica values varying by more than 10 kcal mol^{-1}). [?], [?], [?] In these systems the error obtained more than halves between ensembles of 5 and 25 replicas but after this 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 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. ESMACS protocols 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 tumour sensitivity to tyrosine kinase inhibitors in patients with non-small-cell lung cancer (examples shown in Figure 1). 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

Molecular dynamics is a well established computational methodology for studying the time-evolution and conformational dynamics of a diverse array of physicochemical systems at the molecular level, from which a whole host of physical and chemical properties can be determined [?]. The implementation of any physically realistic molecular simulation has, however, always been an involved and multistage process, often requiring the scientist to overcome a large manual overhead in the construction, preparation, and execution protocols needed 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 theses steps 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 model building (including incorporation of mutations into patient specific protein models); run of ensembles of MD simulations, using a range of free energy techniques; and statistical analysis.

In this work we demonstrate an enhancement of BAC, called high throughput BAC (HT-BAC), which builds upon the Ensemble Toolkit (and thus RADICAL-Pilot) to create a flexible software framework for runtime execution and performance.

III. Computational Challenges at Extreme Scale

Supercomputers and high-performance computing environments were designed to primarily support the execution of single simulations. More precisely, current supercomputers 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). However as the nature of scientific inquiry that HPC are used for, as well as the programs and applications to support that inquiry has evolved, there is a critical need to support the scalable and concurrent execution of a large number of tasks, each of which is typically smaller than the traditional single simulation.

The aggregation of these tasks and their dependencies are often 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 over the years in response to the aforementioned problem. Each workflow system has its strengths and unique capability, however each system typically introduces its own set of 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 workflow system 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 monolithic workflows. In the next section we will discuss how we have designed and implemented the necessary building blocks (RADICAL-Cybertools) that can be used to implement and execute at scale the domain specific workflows that have the necessary properties.

IV. Ensemble Toolkit

We have designed RADICAL-Cybertools to address some of the challenges in developing and executing workflows on HPC platforms.

A. *RADICAL-Cybertools*

We discuss the RADICAL-Cybertools (RCT) software stack used support the flexible and scalable execution of pipelines and upon which HTBAC is built. RCT consists of three main components: Ensemble Toolkit [?] provides the ability to create and execute ensemble-based workflows/applications with complex coordination and communication but without the need for explicit resource management. It uses another RCT component — RADICAL-Pilot [?] which provides resource management and task execution capabilities, which in turn uses RADICAL-SAGA [?], [?] as an interoperability resource access layer.

RCT eschew the concept of a monolithic workflow systems in favor of a fresh perspective to workflows using "building blocks". RCT provide scalable implementations of building blocks in Python and are used to support dozens of scientific applications on high-performance and distributed systems. RCT has also been used extensively to support biomolecular sciences algorithms/methods, e.g., replica-exchange and adaptive sampling.

RADICAL-Pilot: A primary challenge faced is the scalable execution of applications consisting of large ensembles of tasks. Traditionally, each task is submitted as an individual job, or MPI capabilities are used to execute multiple tasks as part of a multi-node single job. The former method suffers from unpredictable queue time for each job; the latter is suitable to execute tasks that are homogeneous and have no dependencies, and relies on the fault tolerance of MPI.

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) to decouple the initial resource acquisition from task-to-resource assignment. Once the pilot (container-job) is scheduled via the LRMS, it can then directly be populated with the computational tasks. This functionality allows all the computational tasks to be executed directly on the resources, without being queued at the LRMS individually. This approach thus supports the requirements of task-level parallelism and high-throughput as needed by science drivers.

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 pattern for ExTASY, implementing two sampling algorithms for execution of both multithreaded and MPI tasks on ARCHER, Stampede and Blue Waters HPC machines. With the same code base, EnTK is being used to develop inversion of full-waveform, wide-bandwidth data for seismic tomography, and a new downscaling methodology to quantify the photovoltaic (PV) energy potential in selected regions of the USA.

Currently, EnTK enables the execution of up to $O(10^4)$ tasks on thousands of CPU and GPU nodes on the resources of XSEDE, ORNL, NCSA, NWSC and UK NSS, for a total of eight HPC machines. 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 four components to the end-user: **Pipeline**,**Stage**, **Task** and ‘Application Manager‘’. These components enable users to create ensemble-based applications and specify target resources for their execution. The ‘Application Manager‘ and ‘WF Processor‘ are responsible for the transformation of the application workflow into a sequence of workloads. Internally the ‘Resource Manager’ and ‘Execution Manager’ components enable the acquisition of resources and the management of execution of these workloads.(see **Figure 2**, right schematic of left panel).

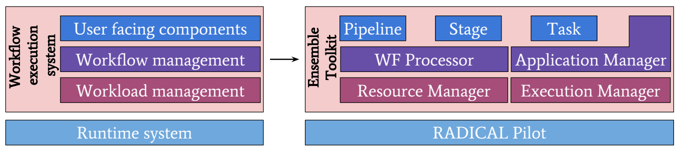

Fig. 2. **Left:** Ensemble Toolkit overview, showing how the abstract workflow execution system is mapped to specific components exposed to the users and components internal to the toolkit

A description of the set of resources to be utilized is provided to the **Application Manager**. This includes properties like walltime, number of nodes and credentials for resource acess. The **Task** component is used to indicate an executable along with its input/output data, resource and environment requirements. The **Stage** component is used to group tasks that have the same dependencies and can be executed concurrently, while the **Pipeline** component is used to describe a sequence of stages, i.e., stages that need to be executed sequentially, not concurrently.

EnTK enables users to program ensemble-based applications directly in Python, without requires a domain-specific language. The use of the Task, Stage, and Pipeline components 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 DAG, a process that imposes a rigid representation model on the domain scientists.

Once the workflow of an ensemble-based application is completely described, one or more pipelines are submitted to the **Application Manager** module for execution. The Application Manager sets up a messaging infrastructure for communication and then processes each pipeline, identifying tasks which have their dependencies satisfied and can be executed concurrently. These tasks are then handed over to the **Execution Manager** that binds tasks to resources on the basis of a set of heuristic-based decisions called "execution strategy". The Execution Manager then 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)

HTBAC is a workflow system that uses RADICAL-Cybertools to implement ESMACS, and similar protocols such as TIES. These workflows consist of pipelines with stages comprised of heterogeneous tasks. For example equilibration and production, followed by post processing steps.

Although ESMACS and TIES methods are similar at a high-level, viz., they are comprised of concurrent, multi-stage pipelines with synchronization. The different protocols supported by HTBAC differ in the details of the pipelines, stages and synchronization [**?**].

HTBAC uses the Ensemble Toolkit (EnTK) API to express these workflows. RADICAL-Cybertools provides advanced resource management capabilities and, thereby delivers the necessary high-throughput capabilities required. The framework is generic and can be simply expanded to run the ESMACS protocol. EnTK provides a common API, execution and programming model to these different methods, and thus will minimize development effort and complexity.

The Ensemble Toolkit API exposes four components (**Resource Handle**, **Pipeline**, **Stage**, and **Task**) to the user which express the application logic of HTBAC. We describe these components for one of the HTBAC protocols, ESMACS. The ESMACS protocol contains a set of pipelines 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 contian heterogeneous workloads. Although each stage of a pipeline depends on its predecessor, the pipelines execute independent of each other. The pattern within pipelines are identical and describes a set of simulations as shown in Fig 3

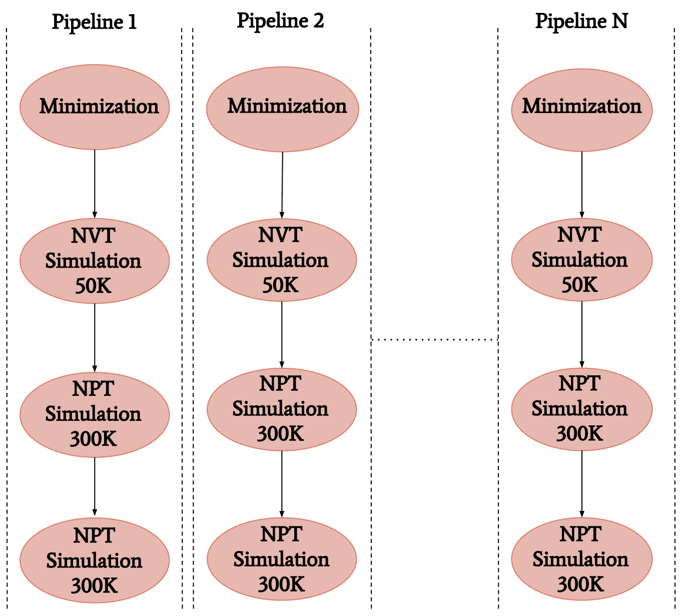

Fig. 3. **ESMACS protocol indicating n-pipelines where each pipeline represents a set of simulations which are captured as stages by EnTK**

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 preliminary and last stages perform staging the input/output data. The middle stages indicate simulation tasks as shown in Fig 3. 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 4.

We define the client resource in Fig 4 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 in a computational campaign that will consume 150M core hours on the NCSA Blue Waters machine, we study the scalability of the ESMACS protocol to determine optimal workflow sizing and resource utilization. The goal is

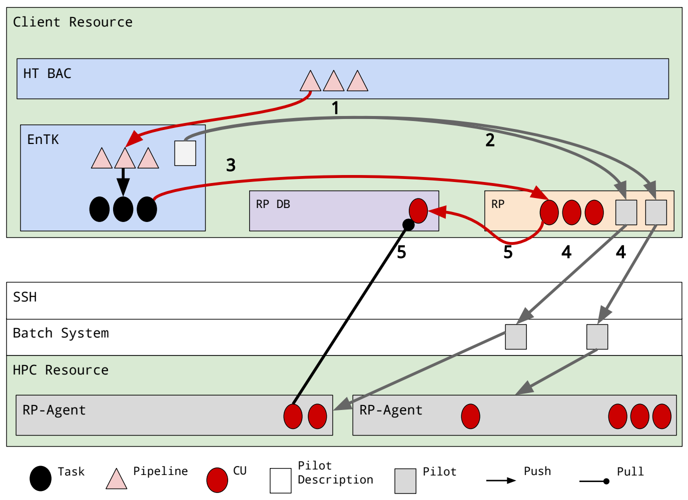

Fig. 4. **Integration between HTBAC workflow and EnTK. Numbers indicate the temporal sequence of execution. RADICAL-Pilot (RP) database (DB) can be deployed on any host reachable from the resources.**

twofold: (1) understanding the invariance of HTBAC execution time over the number of workflow pipelines executed; and (2) studying how EnTK—the RCT tool responsible for coordinating the distributed execution of the protocol—scales its performance in relation to the size of HTBAC workflow.

1) Experiment Design: We designed two experiments to measure HTBAC and EnTK weak scalability when executing an increasing number of concurrent pipelines. According to the use case described in Section V, 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 with the HTBAC workflow and a null payload (/bin/sleep 0). The goal is to isolate the overheads of EnTK from the specifics of: (1) the executables of the HTBAC workflow; (2) the runtime used to execute the workflow on the HPC resources; and (3) 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 HTBAC workflow with the use case payload. The comparison between the two experiments enables performance analysis of EnTK to understand whether and how the size of the executed workflow affects its overheads. Further, Experiment 2 shows also whether HTBAC is sensitive to the number of concurrent pipelines executed.

Both experiments measure the weak scalability of HTBAC and EnTK. 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 both HTBAC and EnTK, it is of limited interest for the current use case. The driving goal of HTBAC is to increase throughput by a means of concurrency, not in the number of sequential execution per core. This is one of the reasons because our project targets large HPC machines instead of so-called HTC infrastructures.

2) Experiment Setup: We perform both Experiment 1 and 2 on the Blue Waters HPC cluster—a 13.3 petaFLOPS Cray at NCSA and University of Illinois, 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. RCTs support the use of both type of architectures and we previously benchmarked the use of GPUs for later evaluation.

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 I shows the setup for Experiment 1 and 2. Currently, the ESMACS protocol is executed with up to 25 concurrent pipelines. This number is suboptimal as pipelines are independent and therefore their concurrent execution does not entail communication overhead. Further, the system simulated can benefit from 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 RADICAL-Pilot 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 use the same number of cores in a pilot as those required by the workload. We use between 64 and 1024 cores in both Experiment 1 and 2, and we always keep the number of simulations equal to the number of cores. We use either 1 or 8 cores per task depending on the task requirements.

All the experiments use Ensemble Toolkit version 0.4.7 and RADICAL-Pilot version 0.42. In the MD workload of the HTBAC workflow, the simulation tasks are executed using 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 both the null and MD workload for two trials at each pipeline configurations.

3) Results: First we characterize the overhead of RADICAL-Pilot (RP) and EnTK in the null workload, where we isolate the overhead introduced by the two systems (Figure 5). 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 on a shared workstation like the one used for our experiments.

RP overhead is also, on average, superlinear but with a much greater variance. This variance is due to mainly two factors:

TABLE I
Experiment 1 executes all 7 stages with the null payload, Experiment 2 uses the actual payload

Experiment ID & Description	Trials	Pipelines	Stages	Tasks	Total Number of Cores
1 (Null)	2	8, 16, 32, 64, 128	7	7	64, 128, 256, 512, 1024
2 (Use-Case)	2	8, 16, 32, 64, 128	7	7	64, 128, 256, 512, 1024

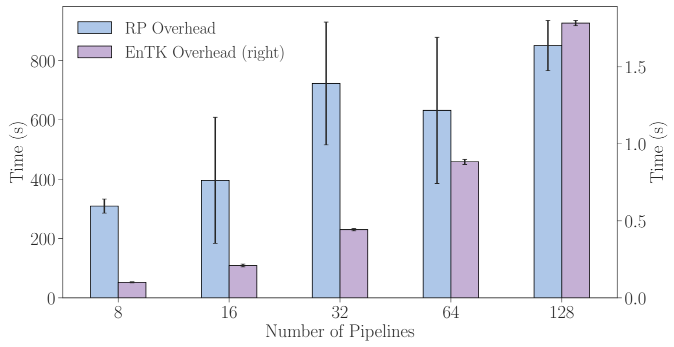

Fig. 5. .

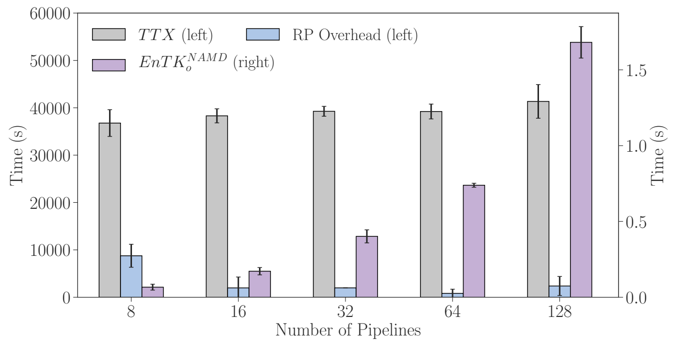

Fig. 6. .

Network latency and filesystem latency on the HPC resource. EnTK submits CU descriptions to the MongoDB used by RP, and RP pilot pulls these descriptions from the same database. As described in Section VI-2, this pull operation occurs between Germany and USA, introducing varying amount of latency. Further, RP pilot writes and reads the CU descriptions multiple time from to and from the shared filesystem of the HPC machine. Together, these two factors introduce delays in the scheduling of the CUs.

As we introduce the use-case NAMD and accessory payload (Figure 5), we show 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 notice the TTX scales weakly across pipelines of different size and that, when accounting for variance, RP overheads also shows weak scaling behavior. As expected, EnTK overhead remains superlinear 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 payload executed. Their scaling behavior and, to some approximation, their absolute values are comparable between Figure 5 and 6. 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

- Importance of Accurate and well bounded time-to-solutions
- Discuss Weak scaling plot (Result)
- Most binding affinity calculations will use differing protocols that will use the same pipeline (future). Discuss (i) implementation of multiple protocols (simplicity and extensibilty), (ii) constant TTC.
- New platforms and new methods (including advanced sampling to guide simulations)

The tools presented in this work represent a vital step towards scaling the use of molecular dynamics based fee energy calculations to the point where they can produce actionable results both in the clinic and industrial drug discovery. The first necessary step is to move beyond the current paradigm of running individual simulations, which provide irreproducible results and cannot provide meaningful error bars. The ability to flexibly scale ensemble simulations and to adapt protocols to the systems of interest is vital if the field is to be able to produce reliable and accurate results on timescales which make it viable to influence real world decision making. In the short term the development of HTBAC will allow us to significantly increase the size of study which is practicable. 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 will we be able to reveal the limits of existing simulation technology and the potentials used to approximate the chemistry of the real systems.

As workflows are increasingly exploratory in nature and adaptive (i.e., evolving) in a way that the full execution (i.e., task graph) is not determinable or discernible in advance, it is important to consider workflows as real time evolution of computational experiment. Thus the flexibilty and reliable performance are concomitant requirement.

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

This work is made possible by an NSF PRAC award for computer time on Blue Waters via NSF OAC 1713749. The middleware and other software capabilities were supported by RADICAL-Cybertools (OAC 1440677) and ”The Power of Many: Ensemble Toolkit” (NSF ICER 1639694). We thank Andre Merzky and other members of the RADICAL team for support. PVC and DWW would like to acknowledge the support of the EU H2020 CompBioMed (675451), EUDAT2020 (654065) and ComPat (671564) projects.