ENABLING ENSEMBLE-BASED METHODS FOR COMPUTATIONAL DRUG CAMPAIGNS AT SCALE ON HIGH PERFORMANCE COMPUTING CLUSTERS

BY JUMANA DAKKA

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Dr. Shantenu Jha, Dr. Matteo Turilli

and approved by

New Brunswick, New Jersey

April, 2019

ABSTRACT OF THE THESIS

Enabling Ensemble-based Methods for Computational Drug Campaigns at Scale on High Performance Computing Clusters

by Jumana Dakka

Thesis Director: Dr. Shantenu Jha, Dr. Matteo Turilli

Free energy calculations that use molecular dynamics (MD) simulations are emerging as an important tool for studying important problems like computational drug design. Recent evidence suggests that free energy calculations, specifically binding affinity calculations, i.e., calculations that quantify the strength of interactions between drug molecules and target proteins, can achieve useful predictive accuracy ($\leq$ 1 kcal/mol) to impact clinical decision making in computational drug design. However, free energy calculations must provide results rapidly and without loss of accuracy. The dual challenge of scaling thousands of concurrent simulations and adaptive selection of favorable simulations based upon feedback from statistical errors and uncertainty need to be tacked. To address these challenges requires advances in algorithms, efficient utilization of supercomputing resources, and software tools that facilitate the scalable and automated computation of varied free energy calculations. This thesis evaluates the requirements of large-scale and adaptive ensemble-based approaches in order to build a software tool designed to enable applications like computational drug campaigns.

In this thesis, we introduce a software tool called the High-Throughput Binding Affinity Calculator (HTBAC), the primary contributions of which are: (i) its ability to apply recent advances in workflow system building blocks to binding affinity calculations, (ii) the ability to execute simulations independent of the simulation software package or supercomputing resources, (iii) enable features of scalability and adaptivity in order to improve resource utilization and scientific results.

Table of Contents

****

Chapter 1

Introduction

1.1 Motivation

Drug discovery and design is immensely expensive with one study putting the current cost of each new therapeutic molecule that reaches the clinic at US$1.8 billion [1]. A diversity of computational approaches, specifically binding affinity calculations which rely on physics-based molecular dynamics (MD) simulations to quantify the strength of interaction between drugs and proteins have been developed [2]. Blind tests show that binding affinity calculations have considerable predictive potential [3, 4]. Recent evidence suggests that binding affinity calculations can achieve useful predictive accuracy of below 1 kcal mol^{-1} [5] and has led to increased interest from the pharmaceutical industry to design computational drug campaigns using binding affinity calculations.

The improvements in predictive accuracy can be attributed to many advances in methodologies and hardware. Specifically, ensemble-based binding affinity calculations, which favor many shorter MD simulation trajectories over few longer MD simulations, have been shown to increase sampling efficiency while also reducing time to clinical insight [6]. For binding affinity calculations to gain traction, they must have well-defined uncertainty and consistently produce statistically meaningful results.

Computational drug campaigns rely on rapid screening of millions of drug candidates, which start with an initial screening of candidates to filter out the ineffective binders before using more sensitive methods to refine the structure of promising candidates. Two prominent ensemble-based binding affinity protocols, ESMACS and TIES [7], have shown the ability to consistently filter and refine the drug design process. The ESMACS (enhanced sampling of molecular dynamics with approximation of continuum solvent) protocol provides an “approximate” endpoint method used to screen out poor candidates. The TIES protocol (thermodynamic integration with enhanced sampling) uses the more rigorous “alchemical” thermodynamic integration approach [8, 9]. These protocols have produced statistically meaningful results for industrial computational drug campaign [10].

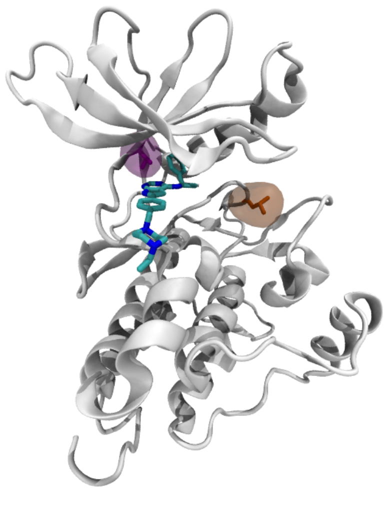
In recent years, considerable effort has been put into improving the efficiency of binding affinity calculations [11, 12, 13]. As computational drug campaign can cover millions of candidates and require hundreds of millions of core-hours for screening, it is important that binding affinity calculations be effective and aim to optimize the accuracy and precision of results. This is challenging as, by definition, drug discovery involves screening drug candidates that are highly varied and potentially unique in their chemical properties. The variability in the drug candidate chemistry results in diverse sampling behavior that contributes to the statistical uncertainty of binding affinity predictions. Further, a particular difficulty comes from the fact that not all changes induced in protein shape or behavior are local to the drug binding site and, in some cases, MD simulations will need to adjust to account for complex interactions between drugs and their targets within individual studies.

Moreover, computational drug campaigns are becoming increasingly important in making individualized therapeutic decisions and to address the concern of drug resistance. Given the rapidly decreasing cost of next-generation DNA sequencing, many cancer centers are sequencing 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, the vast majority of clinically observed mutations are rare [14, 15]. Essentially, this ensures that it will be impossible to make therapeutic decisions about the majority of individual patient tumors by using catalog-building alone.

Moreover, resistance to therapy is becoming increasingly important in drug discovery. In recent years, chemotherapy based on targeted kinase inhibitors (TKIs) has played an prominent role in the treatment of cancer. 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 [16]. 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. 1.1).

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 [17, 18]. Imatinib, the first of these of drugs, is partially credited for doubling survivorship rates in certain cancers [18, 19].

Figure 1.1: Cartoon representation of the Epidermal Growth Factor Receptor (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.



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 to address the need for precision cancer therapy, 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 drug campaigns.

Fortunately, improvements in computational power and algorithm design are enabling the use of simulations to reliably quantify differences in binding strength. This provides the opportunity to use advances in simulations to supplement existing inductive decision support systems with deductive predictive modeling and drug ranking [20, 21].

Computational drug campaigns using ensemble-based approaches rely on rapid screening of millions of drug compounds that require running a large number of simulations in parallel. This effort requires a significant amount of computational resources, beyond standard small-scale computing facilities and instead using scalable platforms like high-performance computing (HPC) facilities. Applications like computational drug campaigns can exploit the availability of HPCs to speed up expensive calculations using free academic resources.

In addition, for ensemble-based approaches to make the necessary impact on computational drug campaigns, two main computational challenges need to be tackled. First, is the challenge of scaling a large number of parallel simulations and second is enabling adaptive selection of simulations using feedback from statistical errors and uncertainty. The second challenge arises on the basis of designing successful drug campaigns that permit the rapid time-to-solution to make clinical impact. As indicated earlier, drug candidates vary by chemical properties, which poses difficulty in identifying when small chemical changes result in large binding strength changes for individual drug candidates. In general, there is no way to know exactly which setup of simulations a particular drug candidate requires before runtime. Traditionally, computational drug campaigns use approximations of simulation parameters in order to guarantee acceptable scientific results for a general set of drug candidates, by executing simulations for a predetermined and invariable duration [22, 23]. This approach has at least two shortcomings: by failing to account for simulation parameters of individual candidates, it potentially wastes valuable computational resources and fails to account for the different values of the simulations results. For example, some drug candidates can reach their optimal solution faster than others and can therefore release their computational resources to allow new simulations to begin executing.

However, this new approach requires monitoring the progress of simulations and enacting on decisions about continued execution on the basis of scientific results, as opposed to running all simulations for a predetermined and invariable duration. The benefit of enabling such feature of enacting upon runtime decision allows computational drug campaigns to utilize resources more efficiently by reallocating resources during runtime to execute only those simulations that yield the fastest time-to-solution or produce the most accurate scientific results for a fixed computational budget. These challenges indicate that computational drug campaigns not only require software tools that enable scalability and management of highly concurrent simulations on HPC, but also require the ability to make runtime decisions based on scientific results generated during execution.

1.2 Objective

The objective of this thesis is to develop a software tool for supporting scalable, adaptive and concurrent ensemble-based approaches tailored to computational drug campaigns. In this thesis we focus on a particular application in computational drug campaigns using binding affinity calculations, yet the framework can apply to other application that entail ensemble-based approaches. The design of the software tool is based on a set of requirements and we evaluate the design using a set of quantitative and qualitative metrics.

To show a successful design of the software, it is necessary to understand the components of the application that motivate the need for such design. It is necessary to understand the limitations and landscape of pre-existing software tools as well as evaluate the user requirements. Lastly, it is necessary to understand how computational resources and similar applications will evolve in the future and how this evolution will impact the design of the framework.

We implement our design using a software tool called HTBAC. We support our design decisions using scientific validation of a case-study in binding affinity calculations to show that the software tool meets identified requirements.

1.3 Overview

The motivation and objective for this thesis is given in Chapter 1. We discuss the impact and key components of computational drug campaigns using binding affinity calculations. We discuss the current landscape of software systems that are used to support computational drug campaigns, and highlight the software barriers that limit scientific progress.

In Chapter 2, we provide background information necessary to appreciate this thesis. First we introduce and motivate ensemble-based approaches for calculating free energy, specifically binding affinity calculations using MD simulations. We also discuss the specific composition of the application that motivates the design of the framework and experiments that evaluate the requirements of the software. Next, we provide an overview of the existing approaches to execute ensemble-based binding affinity calculations, and highlight the computational challenges associated with existing approaches. We also include a section discussing the motivation for adaptivity given requirements of the aforementioned application and discuss the computational limitations of adaptive execution. We include a background section with a discussion of the building block approach in workflow systems, focusing on RADICAL-Cybertools (RCT) that can provide a solution to the aforementioned challenges. Next, we introduce two specific binding affinity protocols that are used in the experiments, namely ESMACS and TIES. Lastly, we include a section provided by our use-case collaborators at University College London (UCL) that describes and validates a set of well-known commercial physical systems used across many computational drug campaigns. These physical system describe the physical and chemical properties of the simulations that are used to conduct the experiments in this thesis.

Chapter 3 describes the design, architecture and implementation of HTBAC. In this chapter we also discuss the implementation of RADICAL-Cybertools–a set of middleware tools that address the computational challenges of scalability and adaptive execution—and describe how they are used by HTBAC to manage the scalable and adaptive execution of binding affinity calculations. We conclude this chapter with an implementation of adaptive and non-adaptive versions of TIES using HTBAC to show the encoding of different applications using HTBAC.

In chapter 4 we present and discuss experiments performed to optimize the performance of HTBAC. We evaluate the weak and strong scaling properties using the ESMACS and TIES protocols, with the physical systems provided by UCL. We also provide experiments conducted by UCL that validate the adaptive requirements of HTBAC, using the TIES protocol.

In Chapter 5, we outline they key conclusions of this thesis with a discussion of the impact of HTBAC. We discuss how well HTBAC satisfies the set of requirements we have identified in chapters 3 and how well we achieved the objectives of this dissertation. Finally we highlight the near-term possible directions for the development of HTBAC.

Chapter 2

Background

In this section we motivate ensemble-based binding affinity calculations, and the computational challenges associated with executing ensemble-based binding affinity calculations in the context of computational drug campaigns. Next, we discuss the landscape of existing software solutions that enable execution of ensemble-based simulations and their limitations. Next, we discuss the building blocks approach to workflow tools, focusing on RADICAL-CyberTools (RCT), that address the aforementioned challenges. Next, we provide the background and composition of two widely-known binding affinity protocols that are be used to perform the experiments using HTBAC. Lastly, we include a brief overview describing the physical systems used in the experiments.

2.0.1 Ensemble-based Binding Affinity Calculations

Binding affinity calculations rely on free energy calculations using MD simulations to quantify the interactions between drug candidates and target proteins. Free energy calculations using MD simulations occur in a wide range of research including protein folding and assessing small molecule binding. Free energy calculations require three main components: (1) suitable Hamiltonian model; (2) sampling protocol; and (3) estimator of free energy. Several approaches to computing binding affinity exist, amongst which relative binding affinity (or binding free energy) calculations are generating accurate predictions, delivering considerable promise for computational drug campaigns [24].

Specifically, ensemble-based MD simulations have been shown to reduce the sampling time required to deliver the precision necessary to meet the requirements of computational drug campaigns. In fact, the lack of reproducibility of single simulation approaches has been show in both HIV-1 protease and MHC systems, with calculations for the same protein-ligand combination, identical initial structure and force field, which produce binding affinities varying by up to 12 kcal mol^{-1} for small ligands [25, 26, 27]. Indeed, these works have revealed how completely unreliable single simulation based approaches are.

Previous work using ensemble-based simulation approaches has also reliably produced results in agreement with published experimental findings [26, 28, 27, 7, 10, 29], and correctly predicted the results of experimental studies performed by different academic groups [30]. While the accuracy of force fields could be a source of error, the very large fluctuations in calculations account for the lion's share of the variance (hence also uncertainty) of the results.

Several ensemble-based approaches are widely used to compute binding free energies, studying different problem spaces. For example, a popular approach is to use Markov state models to learn a simplified representation of the explored phase space and to choose which regions should be further sampled [31]. Replica exchange with solute tempering uses the Metropolis-Hastings criteria to make periodic decisions about what regions of the phase space to sample [32, 33, 34]. In expanded ensemble MD simulations, thermodynamic states are explored via a biased random walk in state space [35]. Approaches that learn by exchanging information have been found to improve sampling results and decorrelate as fast or faster than standard MD simulations.

Using these approaches to compute binding free energy calculations, sampling is performed at discrete regions along the transformation between the two compounds. The choice of where exactly this sampling occurs is a key determinant of the uncertainty in and accuracy of the calculations [36, 37]. Increasing MD simulations with many ensembles in regions of most rapid change reduces errors on the predicated binding affinity.

2.0.2 Computational Challenges

Using ensemble-based approaches to compute binding free energy or simulations in general involves a hierarchy of computational processes. At the lowest level is the specific simulation using an simulation engine. In the case of MD simulations, there are well-known MD engines such as NAMD, Gromacs or AMBER. An ensemble-based **algorithm** (or equivalently **protocol**) is comprised of multiple such MD simulations that are collectively used to compute a scientific measurement In the case of binding affinity calculations, a protocol is used to execute MD simulations that compute the the binding free energy of a single drug candidate. Given that ensemble-based simulations require highly parallel execution, to reduce the simulation time of the simulation, which may require millions of time-steps, domain scientists often take advantage of the capabilities of high performance computers.

There are multiple protocols that can be used, each comes with its specific trade-offs. For example, TIES and ESMACS are two protocols to compute binding affinities that differ in their accuracy but also their computational cost. The computational instance implementing a protocol with specific parameter values, number of simulations and other computational aspects of that protocol, constitutes a **workflow**. A workflow may be fully specified a priori, or it may adapt one or more of its properties, say parameters, as a consequence of intermediate results. Typically, there is a one-to-many relationship between protocols and workflows and different workflows can be used to compute a given binding free energy calculation for a given drug candidate.

When multiple drug candidates need to be evaluated with certain constraints and a defined objective, the entire computational activity (i.e., computing binding affinities for multiple drug candidates) is referred to as a **computational campaign**. The objective of the computational campaign is to maximize the number of drug candidates for which the binding affinity of each individual candidate is determined to within a (given) acceptable level of error. The campaign is constrained by the computational resources available, measured in thousands of core-hours.

Until recently, simulations have generally been limited to at most tens of drug-target combinations. As computational campaigns become larger and more ambitious the inefficiency, and associated cost, of fire-and-forget approaches based on standardized and approximated sampling (e.g. simulations of fixed length) are multiplied. Moreover, strict bounds on time-to-solution exist for clinical insight. This requires both scale and efficient utilization of computational resources. To meet this objective, each workflow computing the binding affinity of a drug candidate is adaptively executed. Adaptive approaches are capable of reducing the computational effort and allowing informed trade-offs between factors such as simulation error tolerance, time-to-solution and computational cost. Executing scalable and adaptive simulations on production-grade HPC resources using ensemble-based methods presents several challenges [22, 23].

The scientific goals require advanced computational capabilities, which pose the following challenges: (1) scaling the adaptive execution of workflows comprised of multiple heterogeneous protocols; (2) developing simple and usable software systems that support adaptive workflows at scale, invariant to the HPC resource or the simulation engine; and (3) implementing different types of adaptivity.

Currently, HPC software ecosystem mostly enable strong and weak scaling of applications composed by a single simulation that requires large amount of parallelism. This ecosystem has instead limited support for the concurrent execution of workflows, especially when composed of multiple heterogeneous simulations. Particularly limiting are the long queue waiting time and the limited amount of concurrency and flexibility offered by machine and architecture-specific tools that enable bulk submission of simulations.

Multiple workflow systems have emerged in response to this and other problems, each with its own strengths and unique capability but also with specific problems and challenges. For example, gSOAP [38] enables web services for HPC applications while Ninf-G [39] and OmniRPC [40] support distributed programming via a client/server architecture. These solutions provide methods to launch applications on remote machines but leave the details of scheduling, resource and data management to the user. On the other hand, domain-specific workflows provide a customized interface to the domain scientist, but require users to manage resource selection and setup the execution environments on the HPC system. Despite the many successes of workflow systems, there is a perceived high barrier-to-entry, integration overhead and limited flexibility.

Moreover, adaptive capabilities can be beneficial to domain science applications but supporting them comes with three main challenges. The first challenges lies in the expressibility of adaptive applications. Composition of adaptive applications requires

Application Programming Interfaces (APIs) that enable the expression of the initial state of the application and specification of algorithms that capture how the application adapts to intermediate data. The former translates to a description of the application Task Graph (TG) while the latter specifies adaptive methods that adapt this TG.

The second challenge is in determining when the adaptation is instantiated. The adaptation is described at the end of the execution of a task wherein a new TG is generated. Different strategies can be employed in the instantiation of the adaptation [41].

The third challenge lies in implementation of the adaptation of the application during runtime. We divide this challenge into three parts: (i) propagation of adapted TG to all components; (ii) consistency of the state of the TG among different components; and (iii) minimal overhead of performing adaptive operations compared to the execution duration of the tasks.

Besides the limitations in existing workflow tools, these challenges also exist within the MD simulation engine packages. MD engines like NAMD provide support for executing highly parallel simulations with respectable performance using programming models like message passing, but lack in providing the capability to construct specific algorithms. Given that the execution strategy is tightly integrated with the MD simulation engine itself, domain scientists cannot easily implement any specialized functionality within the MD simulation engine.

To address these novel applications and scenarios, flexible and efficient ensemble-based software tools are required that are invariant to the simulation engine software or the computational resource. For computational drug campaign application, these tools must provide an interface to encode heterogeneous ensemble-based protocols, enable adaptive execution on the basis of scientific results obtained during runtime, enable scalable execution of heterogeneous ensemble-based protocols that is invariant to the type of protocol, simulation engine or the supercomputing resources.

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 applications like computational drug campaigns.

2.0.3 RADICAL-CyberTools

The RADICAL-Cybertools (RCT) are a set of middleware building blocks that address the challenges in developing and executing workflows on HPC platforms. HTBAC uses two RCT components, mainly the Ensemble Toolkit (EnTK) and RADICAL-Pilot (RP). EnTK provides the ability to create and execute ensemble-based workflows with complex coordination and communication but without the need for explicit resource management. EnTK uses RP as a runtime system which provides resource management and task execution capabilities.

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 HPC and distributed systems [42, 43, 44, 45, 46]. In this thesis we show an understanding of how these components have been used to support the flexible and scalable execution of binding affinity calculations. In Chapter 4 we discuss how RCT play an important role in providing the middleware support to build software tools like HTBAC that can support computational drug campaigns.

2.0.4 Protocols for Computing Binding Affinity Calculations: ESMACS and TIES

In this subsection we provide details about two ensemble-based binding affinity protocols, ESMACS and TIES [10, 7] that are used throughout this work. ESMACS implements absolute free energy methods while TIES implements relative free energy methods. Absolute free energy methods calculate the binding affinity of a *single* drug molecule to a protein, while relative methods calculate the *difference* in binding affinity between two (usually similar in structure) drug molecules. Both protocols are designed to use an ensemble MD simulation approach to enhance the reproducibility and accuracy of standard free energy calculation techniques (MMPBSA [47] in the case of ESMACS and thermodynamic integration [48, 9] in TIES). The use of ensemble averaging allows tight control of error bounds in the resulting free energy estimates.

ESMACS and TIES consists of three main steps: minimization, equilibration and production MD (in its current implementation all MD steps are conducted in the MD engine, NAMD [8]). In practice, the equilibration phase is broken into multiple steps to ensure that the size of the simulation box does not alter too much over the simulation. During these steps, positional constraints are gradually released from the structure and the physical system is heated to a physiologically realistic temperature.

Whilst both protocols share a common sequence of steps, the make-up of the ensemble is different. In ESMACS, an ensemble consists of a set of 25 **replicas**, i.e., identical simulations differing only in the initial velocities assigned to each atom. In TIES, the ensemble contains a set of λ windows, each spawning a set of replicas. As a transformation parameter λ increases from 0 to 1, the system description transforms from containing an initial drug to a target compound via a series of hybrid states. Sampling along λ is then required to compute the difference in binding free energy. In previous studies, TIES has been deployed using 65 replicas, evenly distributed among 13 λ windows. Following the completion of the simulation steps, both protocols require the execution of free energy analysis steps. The detailed composition of ESMACS and TIES protocols is shown in Fig. 2.1.

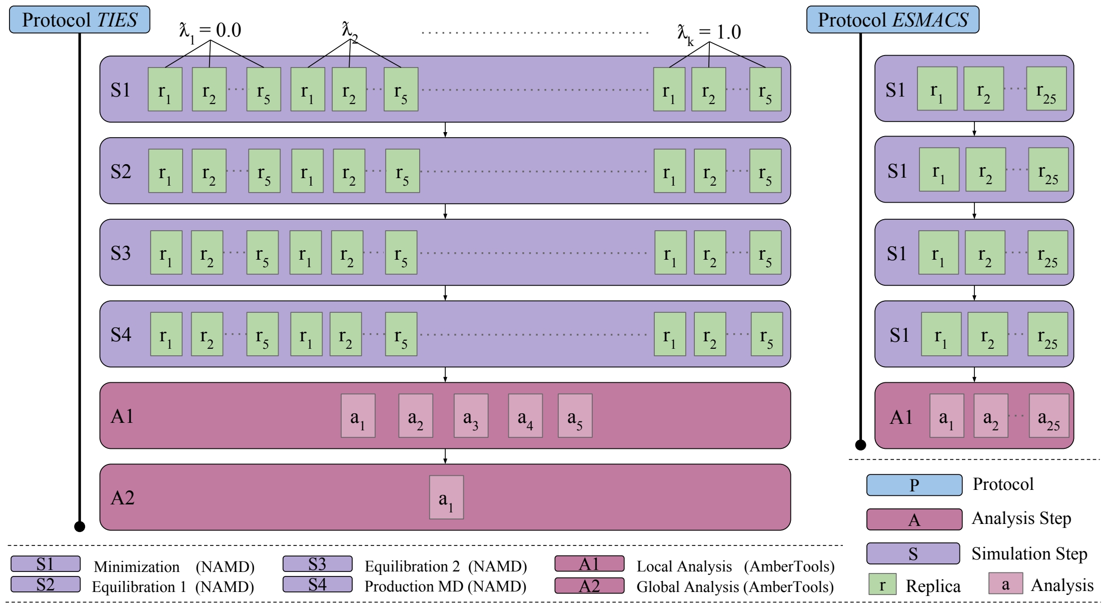
2.0.5 The Value of Adaptivity: A Case-Study

In this section, we motivate the need for adaptivity in computational drug campaigns. Our users of HTBAC presented this particular need given the hindrance on performance and progression of scientific results.

The main driver for adaptivity is that computational campaigns will typically involve compounds with a wide range of chemical properties which can impact the time-to-solution and the type of sampling required to gain accurate results. There may be cases where it is important to increase the sampling of phase space, possibly through expanding the ensemble. In general, there is no way to know exactly which calculation setup a particular system requires before runtime.

Another driver of adaptivity is that, on occasion, alchemical methods may converge very slowly to reach an acceptable error tolerance. This means that the most effective way to gain accurate and precise free energy results on industrially or clinically relevant

Figure 2.1: TIES and ESMACS protocols consist of simulations steps followed by analysis step(s). ESMACS contains 25 replicas per simulation step; TIES contains 5 replicas per λ window. We model TIES with 13 λ windows, spawning 65 replicas in each simulation step. All replicas simulate for 6 nanoseconds (ns).



timescales is to adapt either the workflow corresponding to a specific protocol or adapt different workflows in relation to each other. The latter is referred to as **inter-protocol** adaptivity; the former as **intra-protocol** wherein, for example, the parameter values associated with a specific protocol might change. With thousands of workflows (corresponding to a protocol instances) to adapt in different ways, this has the potential to allow for significant optimization.

In TIES, the change in free energy associated with the transformation is calculated using an adaptive quadrature function which numerically integrates the values of the $\partial U/\partial\lambda$ across the full set of simulated λ windows. Obtaining accurate and precise results from TIES using adaptive quadratures requires that the λ windows correctly capture the changes of $\partial U/\partial\lambda$ over the transformation. This behavior is highly sensitive to the chemical details of the compounds being studied and varies considerably among candidates. Typically, λ windows are evenly spaced between 0 and 1 with the spacing between them set before execution at a distance determined by the simulator to be sufficient for a wide range of systems.

However, the number or the location of the λ windows that will most impact the calculation are not known *a priori*, and varies across candidates. As each window requires multiple simulations, sampling with a high frequency is expensive. Approximations using evenly spaced λ windows reach an acceptable accuracy threshold but adaptive placement of λ windows is likely to better capture the shape of the $\partial U/\partial \lambda$ curve, leading to more accurate and precise results for a comparable computational cost.

In this work, we focus on enabling intra-protocol adaptivity which relies on intermediate runtime results *within* a protocol instance to define the following set of simulations. Instead of approximating the placement of all the λ windows prior to execution, we run TIES with less λ windows and shorter bursts of simulations, analyzing intermediate runtime results (i.e., trajectories) to seed new and ideally placed λ windows.

2.0.6 Physical System Description

This section describes the physical systems that are used in the experiments. These systems are based on real-world examples of common drug candidates provided by our users at UCL and developed in collaboration with GlaxoSmithKline. Scientific and computational improvements require validation across a number of protein ligand complexes. We selected 4 proteins and 8 ligands or ligand pairs to run adaptive free energy calculations. The proteins are the Protein tyrosine phosphatase 1B (PTP1B), the Induced myeloid leukemia cell differentiation protein (MC1), tyrosine kinase 2 (TYK2) and the bromodomain-containing protein 4 (BRD4). Four ligands are alchemical transformations from one to another (used in TIES), four are single ligands suitable for absolute free energy calculations (used in ESMACS). All systems were taken from previously published studies [7].

Simulations were set up using an automated tool developed by UCL called BAC [49]. This process includes parametrization of the compounds, solvation of the complexes, electrostatic neutralization of the systems by adding counterions and generation of configurations files for the simulations. The AMBER ff99SB-ILDN [50] force field was

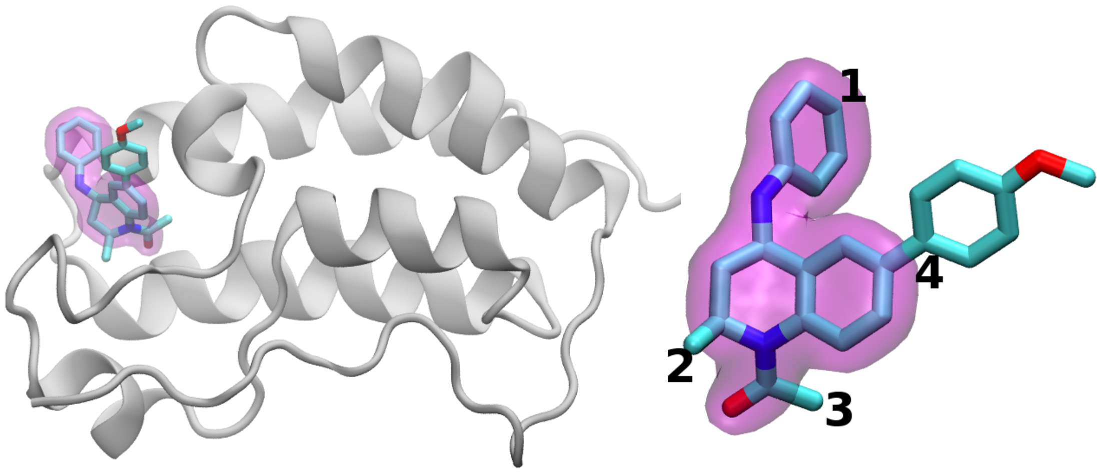


Figure 2.2: (L) Cartoon representation of the Bromodomain-containing protein, BRD4, bound to an inhibitor shown in chemical representation (based on PDB:4BJX). (R) Ligand in cartoon representation with the tetrahydroquinoline scaffold highlighted in magenta. The regions which are modified between ligands investigated are labeled 1 to 4.

used for the proteins, and TIP3P was used for water molecules. Compound parameters were produced using the general AMBER force field (GAFF) [51] with Gaussian 03 [52] to optimize compound geometries and to determine electrostatic potentials at the Hartree–Fock level (with 6-31G** basis functions). The restrained electrostatic potential (RESP) module in the AMBER package [53] was used to calculate the partial atomic charges for the compounds. All systems were solvated in orthorhombic water boxes with a minimum extension from the protein of 14 Å resulting in systems with approximately 40,000 atoms.

Bromodomain-containing proteins (used in ESMACS) as shown in Fig. 2.2, and in particular the four members of the BET (bromodomain and extra terminal domain) family, are currently a major focus of research in the pharmaceutical industry. Small molecule inhibitors of these proteins have shown promising preclinical efficacy in pathologies ranging from cancer to inflammation. Indeed, several compounds are progressing through early stage clinical trials and are showing exciting early results [54]. One of the most extensively studied targets in this family is the first bromodomain of bromodomain-containing protein 4 (BRD4-BD1) for which extensive crystallographic and ligand binding data are available [55].

Chapter 3

HTBAC

In this section we discuss the requirements that HTBAC satisfies, and its design and implementation. In addition, we discuss the execution model which enables adaptive execution of ensemble-based binding affinity protocols. We conclude with examples of adaptive and non-adaptive ensemble-based binding affinity protocols that are described using HTBAC.

3.1 Requirements

HTBAC is a software system for running ensemble-based binding affinity protocols adaptively and at scale on HPC resources. Currently, HTBAC supports protocols composed of an arbitrary number of analysis and simulation steps, and relies on the ensemble management system and runtime system provided by the RADICAL-Cybertools (RCT). HTBAC is designed to be extended to support more types of protocols and alternative runtime middleware.

HTBAC satisfies three main requirements: (1) enable the scalable execution of concurrent protocols; (2) abstract protocol specification from execution and resource management; and, (3) enable adaptive execution of protocols.

Computational drug campaigns increasingly depend on scalable ensemble-based binding affinity protocols. This poses at least two major computational challenges. First, ensemble-based binding affinity protocols require execution coordination and resource management among ensemble members, within protocols as well as across protocols. Second, the setup of execution environments and data management has to preserve efficient resource utilization. These challenges need to be addressed by HTBAC as well as the underlying ensemble management and runtime system.

Adaptive execution of protocols requires the ability to change the control logic of the ensemble execution, based on intermediate results of the ongoing computation. Thus, HTBAC has to support resource redistribution, according to the logic of the adaptive algorithms, enabling the optimization of computational efficiency.

Finally, usability plays an important role in the development of HTBAC. HTBAC has to provide a flexible interface which enables users to easily scale the number of drug candidates and quickly prototype existing and novel binding free energy protocols.

3.2 Design and Implementation

HTBAC exposes four constructs to specify free energy protocols: Protocol, Simulation, Analysis, and Resource. **Protocol** enables multiple descriptions of protocol types, while **Simulation** and **Analysis** constructs specify simulation and analysis parameters for each protocol. **Resource** allows to specify the amount of resources needed to execute the given protocols. Together, protocol instances, simulation and analysis parameters, and resource requirements constitute an HTBAC **application**.

Each protocol models a unique protein ligand physical system. Protocols follow a sequence of simulation and analysis steps, assigning ensemble members to execute independent simulations or analysis. An ensemble member that executes a simulation within a simulation step is referred to as a replica. Each simulation is assigned a different initial velocity, which enables simulations to begin in different parts of the ligand's phase space.

Individual simulations or analyses with input, output, termination criteria and dedicated resources are designed as a computational **task** [56]. Aggregates of tasks with dependencies that determine the order of their execution constitute a **workflow**. In this way, HTBAC encodes N_P instances of the P^{th} protocol as a workflow of computational tasks.

Fig. 3.1 shows the components and subcomponents of HTBAC. The API enables users to describe protocols in terms of protocol type, simulation and analysis steps, and compute infrastructure requirements. The Descriptor component uses two subcomponents to aggregate protocol descriptions into a single application and resource description. Note that Descriptor can aggregate different types of protocols, with different computing and resource requirements.

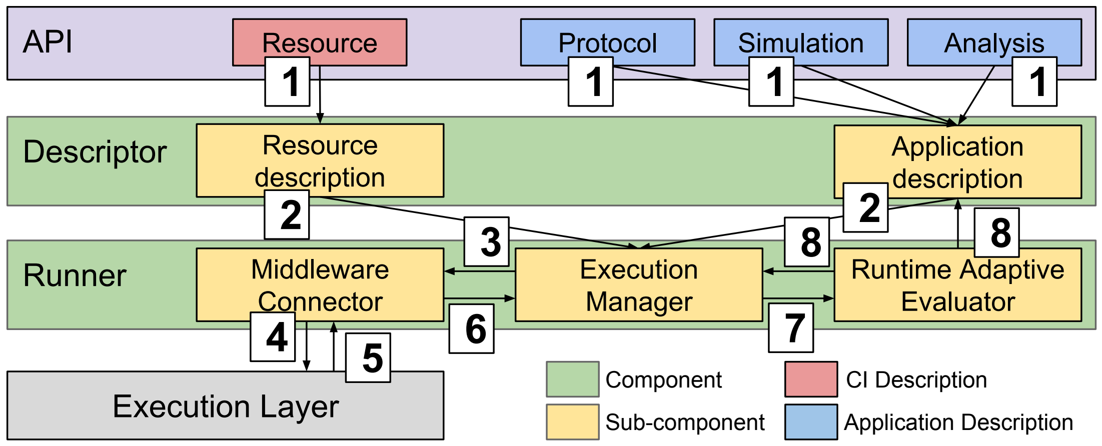


Figure 3.1: HTBAC architecture. Users specify protocol(s) with multiple simulation and analysis steps. Descriptor derives a single application that Runner executes on an external execution layer. Runtime Adaptive Evaluator enables the execution of adaptive protocols.

The Runner component has three subcomponents: Execution Manager, Middleware Connector and Runtime Adaptive Evaluator. The Execution Manager communicates with the execution layer via a connector to coordinate the execution of the application. In principle, HTBAC can use multiple connectors for diverse middleware to access different computing infrastructures.

The Middleware Connector converts the application description of HTBAC into a middleware-specific format. The Execution Manager can pass the given application to the connector in full or only in parts. This enables to start the execution of an application before its full description is available or to change those parts of the application that still have to be executed. This will enable future capabilities like, for example, to concurrently execute the application on diverse middleware.

The Runtime Adaptive Evaluator enables the execution of adaptive applications. This subcomponent can evaluate partial results of an application execution via tailored algorithms. On the base of this evaluation, the Runtime Adaptive Evaluator can decide to return the control to Execution Manager or modify the description of the application that is executed. In this way, HTBAC implements adaptivity for diverse protocols, allowing users to define arbitrary conditions and algorithms.

HTBAC is implemented in Python as a domain-specific library. All components of HTBAC are implemented as objects that communicate via method calls. HTBAC uses two RCT software tools as building blocks [57]: Ensemble Toolkit (EnTK) and RADICAL-Pilot (RP).

EnTK provides HTBAC with the capabilities to execute ensemble-based applications [56]. EnTK exposes three constructs: **Task**, **Stage** and **Pipeline**. Tasks contain information regarding an executable, its software environment and its data dependencies. Stages are sets of tasks without mutual dependencies that can execute concurrently. Pipelines are lists of stages, where stages can execute only sequentially. Pipelines can execute independently. HTBAC uses a Middleware Connector for EnTK to encode a protocol instance as a single pipeline that contains stages of individual simulations and analyses tasks.

EnTK is designed to be coupled with different runtime systems. In this paper, EnTK uses RP to execute tasks via pilots. RP supports task-level parallelism and high-throughput by acquiring resources from a computing infrastructure and scheduling tasks on those resources for execution. RP uses RADICAL-SAGA to interface with several resource managers, including batch systems like SLURM, PBS (pro), and Lustre File System (LSF). Pilot systems execute tasks directly on the resources, without queuing them on the infrastructure's scheduler.

3.3 Execution Model

Users describe one or more protocols alongside their resource requirements via HTBAC's API. The Descriptor component takes these descriptions as input and returns an application description (Fig. 3.1.1). As seen in §3.2, this application consists of a set or sequence of tasks with a set of resource requirements for their execution.

The application description is passed to the Execution Manager of the Runner component (Fig. 3.1.2). The Execution Manager evaluates the resource requirements, selects a suitable connector (currently only to EnTK), tags each protocol instance of the application with an ID, and passes all or part of the application description to the connector for execution (Fig. 3.1.3).

The Middleware Connector of the Runner component gets the application description, converting it into a middleware-specific description (EnTK pipelines of stages of tasks) and a resource request. The connector submits this request to the underlying execution layer (Fig. 3.1.4) and initiates the execution of the application once the execution layer communicates the availability of the resources (Fig. 3.1.5).

The resource requirements specified via HTBAC's API include walltime, cores, queue, and user credentials. EnTK derives a resource request from these requirements, converting it into a pilot description for RP. RP converts this pilot requests into a batch script that is submitted to the specified HPC machine. Once the pilot becomes active, EnTK identifies those application tasks that have satisfied dependencies and can be executed concurrently. EnTK's own Execution Manager uses RP to execute those tasks on the pilot's resources.

HTBAC allows to specify conditions tailored to individual simulation steps of a protocol implementation. We leverage this ability to implement adaptivity by enabling the user to partition protocols into simulation steps and generate new simulation steps at runtime, based on a set of predefined conditions. The user specifies these conditions in an analysis script for the Runtime Adaptive Evaluator subcomponent.

Execution Manager can retrieve the results of simulations (Fig. 3.1.6) and these results can be evaluated by the Runtime Adaptive Evaluator via a user-defined analysis script (Fig. 3.1.7). Depending on the result of the evaluation, the Runtime Adaptive Evaluator may generate new simulation steps, adding them to the application description (Fig. 3.1.8a) or return control back to the Application Manager (Fig. 3.1.8b) without changing the application. If new simulations are to be generated, the Execution Manager bypasses termination of the application, and passes the added application description to the connector. Execution Manager uses the protocols' IDs to keep track of those protocols that require additional simulations at runtime. In this way, older simulations can continue executing while new simulations can be passed to the Execution Manager.

In an adaptive scenario, as the number of simulations grows at runtime, the ratio of cores-to-task fluctuates. EnTK's Execution Manager automatically redistributes an even share of the total requested cores to each simulation. RP allows for new simulations to execute within the pilot's wall-time, without having to acquire new resources via the resource management system.

3.4 RADICAL-Cybertools

In its current implementation, HTBAC uses two RCT components, RADICAL-Pilot as the runtime system and Ensemble Toolkit as the ensemble management system. We have designed RCT to be functionally delineated building blocks in developing and executing workflows on HPC platforms.

EnTK provides the ability to create and execute ensemble-based workflows with complex coordination and communication but without the need for explicit resource management. EnTK uses RP as a runtime system which provides resource management and task execution capabilities. In this section we discuss details of RP, and EnTK, understanding how these components have been used to support the flexible and scalable execution of tools like HTBAC.

3.4.1 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 message-passing interface (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 [58] 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 scientific applications, 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.

3.4.2 Ensemble Toolkit

Ensemble Toolkit (EnTK), 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 (directly with EnTK's own API or to build a higher level of abstraction like HTBAC), programming ensemble-based applications by describing what, when and where should be executed. EnTK uses a runtime system, like RADICAL-Pilot, to acquire the resources needed by applications to manage task execution.

As indicated earlier, EnTK enables the creation of ensemble-based applications by exposing an API with four components: **Application Manager**, **Pipeline**, **Stage** and **Task**.

The use of the Task, Stage, and Pipeline components, implemented as set and list data structures, avoids the need to express explicitly relationship among tasks. These relationships are insured by design, depending on the formal properties of the lists and sets used to partition tasks into stages and group stages into pipelines. Further, EnTK enables an explicit definition of pre and post conditions on the execution of tasks, 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 [56].

As indicated earlier, the **Application Manager** component of EnTK enables users to specify target resources for the execution of the ensemble-based application. This includes properties like walltime, number of nodes and credentials for resource access. Users can also define execution setup parameters such as the number of processes or messaging queues that should be used by EnTK. This allows to size and tune the performance of EnTK, depending on the number of tasks, stages and pipelines, but also on the resources available to the toolkit.

The Application Manager along with the **WF Processor** is responsible for the transformation of the application workflow into workloads, i.e., set of tasks, that can be submitted to the indicated resources for execution. Internally, EnTK's **Resource Manager** and **Execution Manager** components enable the acquisition of resources and the management of execution of these workloads.

3.5 Implementing ESMACS and TIES in HTBAC (Use-Cases)

In §2.0.4 we define the structure of the ESMACS and TIES protocols. Here we provide skeletons of the TIES protocol implemented in HTBAC. In L. 3.1 we show a customization of a production MD simulation step.

```
import htbac.protocols.TIES
ties_1 = Protocol(system = 'brd4-1')
sim = Simulation(name = 'Production MD')
        ... # define simulation conditions
sim.ensemble('replica', range(5))
sim.ensemble('lambda', range(13))
# add simulation configurations to protocol
TIES.step0 = sim
ties_1.append(TIES.step0)
# assign resources, append protocol to Runner
```

```
runner = Runner(resource = 'ncsa.blue_waters',
                walltime = 60)
runner.add_protocol(protocol = ties_1)
# launch application
runner.run()
```

Listing 3.1: TIES protocol implemented with HTBAC. We import the predefined protocol 'TIES'. We assign the physical system to the protocol, we instantiate a simulation, customize its steps (`replica`, `lambda`) and assign it to the TIES's `step0`. We instantiate the Runner with a resource request and pass the protocol description to it.

In §3.3 we show HTBAC's adaptive execution capabilities. In L. 3.2 we provide an intra-protocol adaptive implementation of TIES, based on the use-case of §2.0.5.

```
# we start with the same previous implementation
# provide runner with two flags
runner.run(save_output = True, terminate = False)
# specify adaptivity script
requested_lambdas = AdaptiveQuadrature()
sim1 = Simulation(name = 'production MD 2')
        ...
sim1.ensemble('replica', range(5))
sim1.ensemble('additional lambdas',
    requested_lambdas)
TIES.step1 = sim1
ties_1.append(TIES.step1)
runner.add_protocol(protocol = ties_1)
runner.run(terminate = True)
```

Listing 3.2: Adaptive TIES protocol implemented with HTBAC. Assuming L. 3.1, we run the Runner retrieving runtime results, we specify an adaptivity script for the evaluator, create TIES `step1`. The analysis script operates on partial simulation results, generating new simulation conditions for the next simulation step.

Chapter 4

Experiments

In in chapter we present and discuss experiments performed to optimize the performance of HTBAC. We evaluate the weak and strong scaling properties using the ESMACS and TIES protocols, with the physical systems provided by UCL. We also include a section that validates the adaptivity requirements of HTBAC, using scientific experiments conducted by UCL with the TIES protocol.

Typically, a computational campaign for drug discovery explores a large number of drug candidates by running several workflows multiple times, each requiring thousands of concurrent simulations. Often these drug campaigns require hundreds of million core-hours on HPCs. Therefore, we perform experiments to characterize the weak and strong scaling performance of HTBAC and its overheads using one HPC resource: NCSA Blue Waters. We validate the results of the free energy calculations produced using HTBAC against published results.

Given that protocols like TIES are more computationally demanding than protocols like ESMACS, it is paramount to use resources efficiently, especially for campaigns that have a predefined computational budget. As described in 3, adaptive simulation methods have the potential to reduce the number of simulations without a loss in accuracy and with a lower computational load. Using HTBAC, our collaborators at UCL validated the adaptivity requirements of HTBAC using an adaptive implementation of TIES and measured the improvements in terms of accuracy, reduction of total simulations and computational load.

Table 4.1: Parameters of scalability experiments.

ID	Type of Experiment	Physical System(s)	Protocol(s)	No. Protocol(s)	Total Cores
1	Weak scaling	BRD4	ESMACS	(2, 4, 8, 16)	1600, 3200, 6400
2	Weak scaling	BRD4	TIES	(2, 4, 8)	4160, 8320, 16640
3	Weak scaling	BRD4	ESMACS + TIES	(2, 4, 8)	5280, 10560, 21120
4	Strong scaling	BRD4	TIES	(8, 8, 8)	16640, 8320, 4160
5	Strong scaling	BRD4	ESMACS	(16, 16, 16)	6400, 3200, 1600
6	Strong scaling	BRD4	ESMACS + TIES	(20, 20, 20)	22120, 10560, 5280

4.1 Experiment Setup

Table **??** shows 6 experiments we designed to characterize the behavior of HTBAC on NCSA Blue Waters. Each experiment executes the ESMACS and/or TIES protocol for different physical systems. Experiments 1–6 use the BRD4 physical system provided by GlaxoSmithKline.

Experiment 1 and 2 measure the weak scaling of HTBAC using the ESMACS and TIES protocols. Experiments 3 uses both the TIES and ESMACS protocols, characterizing the weak scaling of heterogeneous protocol executions. Experiments 4 and 5 measure the strong scaling of HTBAC using a fix number of instances of the ESMACS and TIES protocols. Experiments 6 uses both the TIES and ESMACS protocols, characterizing the strong scaling of heterogeneous protocol executions.

In each weak scaling experiment (1–3), we keep the ratio between resources allocated and protocol instances constant. Consistently, for each experiment we progressively increase both the number of cores (i.e., measure of resource) and the number of protocol instances by a factor of 2. In each strong scaling experiment (4–6), we change the ratio between resources allocated and the number of protocol instances: we fix the number of protocol instances and reduce the number of cores by a factor of 2.

Weak scaling experiments provide insight into the size of the workload that can be executed in a given amount of time, while strong scaling experiments show how the time duration of the workload scales when adding resources. For all the weak and strong scaling experiments we characterize the overheads of HTBAC, EnTK and RP, and we show an approximation of the time taken by the resources to become available.

This offers insight about the impact of HTBAC and its runtime system on the time to completion of each workload. In [59], we show baseline performance of HTBAC using ESMACS with a null workload.

For weak and strong scaling experiments, we reduced the number of time-steps of the protocols and omitted the analysis steps $S5$ and $S6$ of their workflows (Fig. 2.1). These simplifications are consistent with characterizing scalability performance instead of simulation duration. The time-steps are set to enable the physical systems to reach steady-state. For the experiments 1–6 we used the following time-steps: $S1 = 1000$; $S2 = 5000$; $S3 = 5000$; and $S4 = 50000$.

We measure the following durations for Experiments 1–6:

- **Total Task Execution Time**: Time taken by all the task executables to run on the computing infrastructure.
- **HTBAC Overhead**: Time taken to instantiate HTBAC, and validate and process the application description.
- **EnTK and RP Overhead**: Time taken by EnTK and RP to manage the execution of tasks.
- `aprun` **Overhead**: Time taken by `aprun` to launch tasks on Blue Waters.

Note that once RP relinquishes the control flow to `aprun`, the precise time at which `aprun` schedules each task on a compute node and the MD kernel of each task begins execution cannot be measured. Instead, for each task, we measure the difference between the task execution time and its `NAMD` kernel execution time, provided by the `NAMD` output logs. In this way, we approximate the time taken by `aprun` to launch the task. Once aggregated, these measures constitute what we defined as `aprun` Overhead. The summation of all durations provides the average wall-time of the pilot job.

We performed all the experiments on Blue Waters, a 26868 node Cray XE6/XK6 SuperComputer with peak performance of 13.3 petaFLOPS managed by NCSA. Consistent with NCSA policies, we initiated the experiments from a virtual machine outside NCSA, avoiding to run persistent process on the NCSA login node. We used HTBAC

0.1, EnTK 0.6, and RP 0.47 and the `NAMD-MPI` MD kernel, and launched via the `aprun` command. For the analysis stages in the TIES protocol we used `AmberTools`.

NCSA sets a system policy on the maximum number of processes that `aprun` can spawn, limiting the number of concurrent tasks we can execute on Blue Waters to $\approx$450. During the execution of Experiment 2, we observed failing tasks with 8 TIES protocol instances, i.e., 520 concurrent tasks. In a trial of 10 repetitions at this scale, we observed an average of 70 ± 6.67 failing tasks. More data would be required to model the distribution type of these results.

NCSA allows to run only one MPI application for each compute node. Thus, we run each MD simulation with 32 cores (i.e., one compute node) even if our performance of `NAMD` on Blue Waters indicated that 16 cores offers the best trade-off between computing time and communication overhead.

4.2 Weak Scaling Characterization

Fig. 4.1(a) shows the weak scaling of HTBAC with the TIES protocol. Each instance of the TIES protocol contains a single pipeline with 4 stages and 65 concurrent tasks. We increase the number of protocol instances linearly, between 2 and 8. When scaling to 8 protocol instances, we execute more than 450 concurrent tasks, the average limit supported by `aprun`, as described in §4.1. This introduces some failures that contribute to a slight degradation in performance.

Fig. 4.1(b) shows the weak scaling of HTBAC with the ESMACS protocol. We increase the number of instances linearly, between 2 and 16. Each ESMACS protocol contains 1 pipeline with 4 stages and 25 concurrent tasks.

Fig. 4.1(c) shows the weak scaling of HTBAC with instances of both TIES and ESMACS protocols. Also in this case, we scale the instances of both protocols linearly, between 2 and 8. The first configuration shows 1 ESMACS and 1 TIES protocol, and with each increase in scale we double the number of protocols. Experiments 2 and 3 show scaling ranges within the limit of the maximum number of concurrent tasks we can successfully execute on Blue Waters.

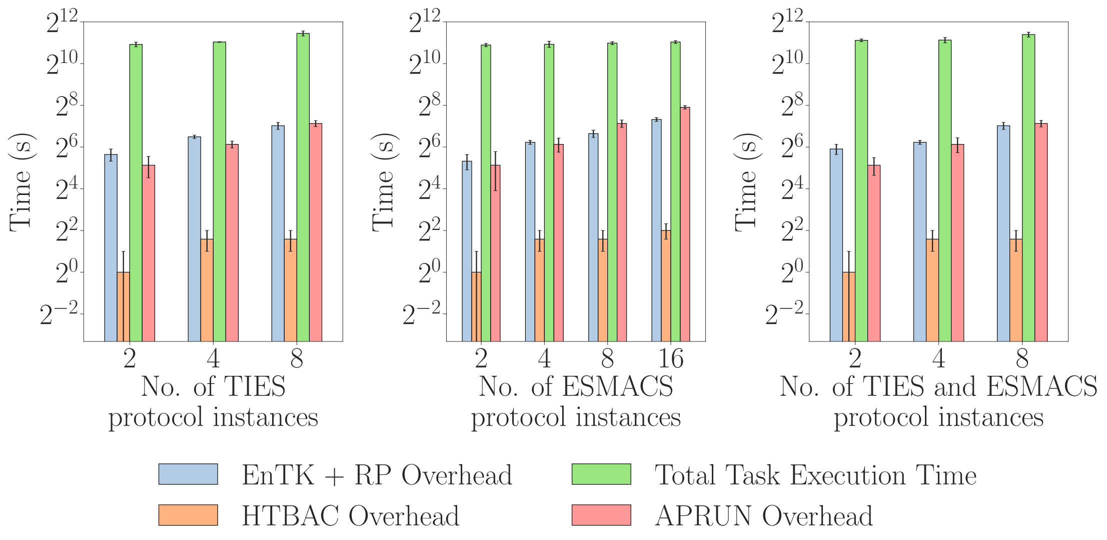


Figure 4.1: Weak scaling of HTBAC. The ratio number of protocol instances to resources is constant. Task Execution Time, and HTBAC, EnTK+RP, `aprun` overheads with (a) TIES (Experiment 1), (b) ESMACS (Experiment 2), and (c) TIES and ESMACS (Experiment 3).

For all weak scaling experiments (1–3) we use physical systems from the `BRD4-GSK` library with the same number of atoms and similar chemical properties. The uniformity of these physical systems ensures a consistent workload with insignificant variability when characterizing their performance under different conditions.

In all weak scaling experiments (Fig. 4.1) we observe that the value of Total Task Execution Time (green bar) shows minimal variation as the number of protocol instances increases, suggesting that HTBAC is invariant to the protocol. We conclude that HTBAC shows near-ideal weak scaling behavior under these conditions.

The HTBAC overhead depends mostly on the number of protocol instances that need to be generated for an application. This overhead shows a super linear increase as we grow the number of protocol instances, but the duration of the overhead is negligible when compared to Total Task Execution Time.

The `aprun` overhead increases as we approach the limit of concurrent `aprun` processes that can be executed on Blue Waters. For example, when scaling to 8 TIES protocol instances (Fig. 4.1(a)), we see that the increase in `aprun` overhead occurs due to task failure. This is explained by noticing that attempts to relaunch failed tasks require additional communication among the nodes that were running the tasks and the MOM Nodes from which the execution is coordinated.

EnTK and RP overheads mostly depend on the number of tasks that need to be translated in-memory from a Python object to a task description [59, 60]. As such, those overheads are expected to grow proportionally to the number of tasks, as observed in Fig. 4.1, blue bars.

The RP overhead is calculated by measuring and aggregating the execution time of the RP components that manage and coordinate the execution of the protocol instances. Among these components, the task scheduler of RP introduces the largest overhead. Due to the general scheduling algorithm loaded by default in RP, the task scheduling overhead scales linearly with the number of tasks that need to be scheduled.

In comparison to Total Task Execution Time, the EnTK and RP overheads are an order of magnitude shorter, yet they directly contribute to the total duration of the application execution. Based on Fig. 4.1, we approximate the use of our systems will results in $\approx 15\%$ additional usage of resource allocation. This overhead can be substantially reduced by using a special-purpose scheduler for RP as illustrated in Ref. [60].

4.3 Strong Scaling Characterization

In Experiment 4 we fix the number of instances of the TIES protocol to 8 (due to the described `aprun` limitations) and we vary the amount of resources between 4160, 8320 and 16640 cores. Assuming the definition of 'generation' in §4.1, given 4160 cores, we can execute 4 generations of 130 concurrent tasks; with 8320 cores, 2 generations of 260 tasks; and with 16640 cores, 1 generation of 520 tasks.

In Experiment 5 we fix the number of instances of the ESMACS protocols to 16 and vary the amount of resources between 3200, 6400 and 12800 cores. In this way, we obtain the same number of generations as in Experiment 4.

In Experiment 6 we fix the number of instances of the ESMACS and TIES protocols to 16 and 4 respectively, and vary the amount of resources between 5280, 10560 and 22120 cores. In this way, we obtain the same number of generations as in Experiment 4 and 5.

Fig. 4.2 shows a linear speedup in Total Task Execution Time for both experiments, proportional to the increase in the number of cores. The availability of more resources for a fixed number of protocols explains this behavior. Overheads remain essentially constant for both experiments when increasing the number of cores. The scheduling of the number of tasks, as opposed to the amount of resources, is the main driver of EnTK and RP overheads (Ref. [60]).

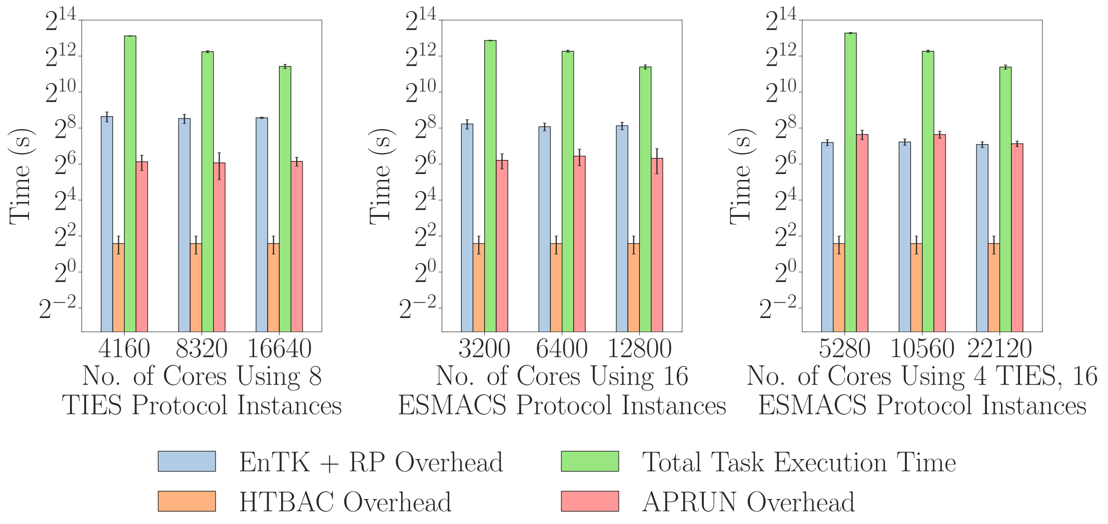


Figure 4.2: Strong scaling of HTBAC. The number of protocol instances is fixed while the number of cores increases. Task Execution Time, and HTBAC, EnTK+RP, `aprun` overheads with (a) TIES (Experiment 4), ESMACS (Experiment 5) and ESMACS + TIES (Experiment 6).

4.4 Validation

In order to validate the correctness of the results produced in Experiment 1–6, using HTBAC and the BRD4-GSK physical systems, we compare our results with those previously published in Wan et al. [10]. In this way, we can confirm that we calculated the correct binding free energies values.

Table 4.2: Validation of HTBAC results against published and experimental values

System	HTBAC (kcal mol^{-1})	Wan et al. (kcal mol^{-1})	Experiment (kcal mol^{-1})
BRD4 **3 to 1**	0.39(10)	0.41(4)	0.30(9)
BRD4 **3 to 4**	0.02(12)	0.01(6)	0.00(13)
BRD4 **3 to 7**	−0.88(17)	−0.90(8)	−1.30(11)

We validated our implementation selecting a subset of the protein ligand systems used in Wan et al. [10]: ligand transformations 3 to 1, 4, and 7. We then performed a full simulation on all 3 systems and calculated the binding affinity using HTBAC.

The results of our experiments, collected in Table 4.2, show that all three $\Delta\Delta$G values are within error bars of the original study, validating the results we produced with HTBAC.

4.5 Validation of Adaptive Requirements

This section presents experiments that validate the adaptivity requirements of HTBAC designed by UCL. Table 4.3 shows 2 adaptivity experiments to characterize nonadaptive and adaptive simulation methods using the TIES protocol. These experiments were also performed on NCSA Blue Waters and utilize the PTP1B, MC1, and TYK2 physical systems.

The adaptivity experiments compare the accuracy and time to solution of non-adaptive and adaptive simulation methods. For the nonadaptive simulation method of Experiment 1 our users preassigned 13 approximated λ windows, consistent with the value reported in Ref. [7]. In this way, this produces 65 concurrent simulations for stages $S1$–$S4$ of TIES (see Fig. 2.1). The production simulation stage $S4$ executes each simulation for 4 ns. Stage $S5$ has 5 analysis tasks which aggregate the simulation results of $S4$. The global analysis stage $S6$ has a single task that aggregates the results from $S5$.

In the adaptive implementation 4.3, the TIES protocol is initialized with 3 λ windows, obtaining 15 replicas, while stage $S4$ of each TIES replica is separated into 4 sub-stages. Each sub-stage runs a 1 ns simulation, followed by an adaptive quadratures analysis which estimates free energy errors with respect to each interval of two λ values.

Experiment 3 compares the adaptive and non-adaptive execution of TIES. Using 65 simulations derives 13 equally spaced λ windows to calculate the free energy with high accuracy. This creates a baseline against which to compare the adaptive and non-adaptive results.

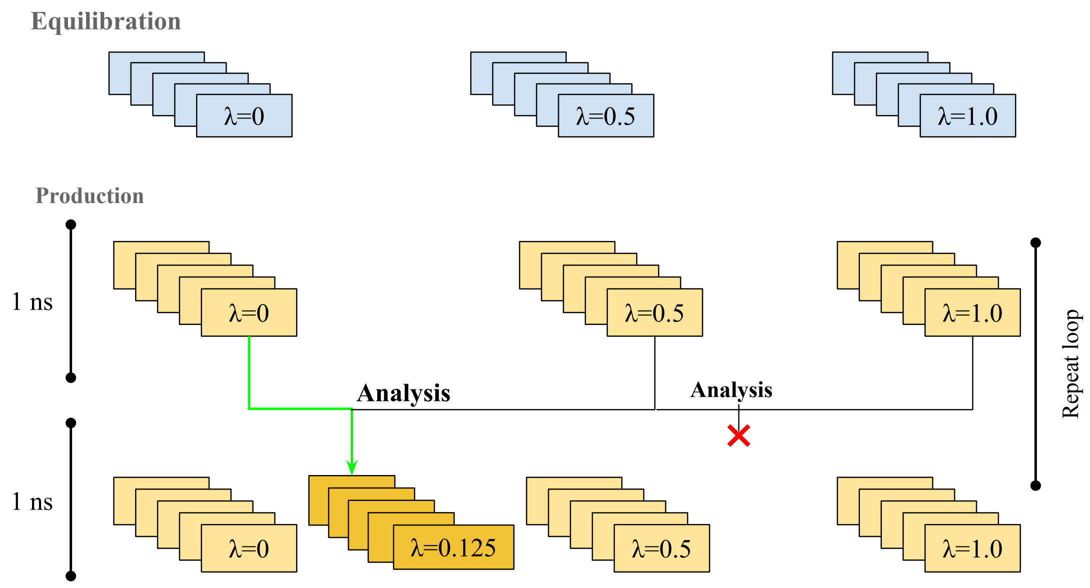


Figure 4.3: Adaptive workflow for TIES. After equilibrating 3 λ, the first stage starts. This is followed by analysis at every λ interval, to decide whether to add a new window in between. In our implementation, the simulation-analysis cycle is repeated for 4 simulation steps, not shown here.

Experiment 1 and 3 are assigned the following simulation time-steps in $S1 = 3000$; $S2 = 50000$; $S3 = 50000$; and $S4 = 2000000$. The adaptive simulation of Experiment 8 uses the same time-steps, apart from $S4$ which is divided into 4 sub-stages of 500000 time-steps each.

The design of HTBAC permits enhancing protocols while continuing to use "static" simulation engines. Our UCL users implemented two adaptive methods using HTBAC:

Table 4.3: Parameters of adaptivity experiments.

ID	**Type of Experiment**	**Physical System(s)**	**Protocol(s)**	**No. Protocol(s)**	**Total Cores**
1	Non-adaptivity	PTP1B, MC1, TYK2	TIES	(1, 1, 1)	2080, 2080, 2080
2	Adaptivity	PTP1B, MC1, TYK2	TIES	(1, 1, 1)	2080, 2080, 2080
3	Reference	PTP1B, MC1, TYK2	TIES	(1, 1, 1)	10400, 10400, 10400

adaptive quadrature and adaptive termination. Both of these methods use the features of adaptivity offered in HTBAC to scale to large number of concurrent simulations and to increase convergence rate and obtain more accurate scientific results.

The aim of introducing adaptive quadrature for alchemical free energy calculation protocols (e.g., TIES) is to reduce time to completion while maintaining (or increasing) the accuracy of the results. Time to completion is measured by the number of core-hours consumed by the simulations. Accuracy is defined as the error with respect to a reference value, calculated via a dense λ window spacing (65 windows). This reference value is used to establish the accuracy of the non-adaptive protocol (which has 13 λ windows) and the adaptive protocol (which has a variable number of λ windows, determined at run time).

One of the input parameters of the adaptive quadrature algorithm is the desired acceptable error threshold of the estimated integral. We set this threshold to the error of the non-adaptive algorithm calculated via the reference value. The algorithm then tries to minimize the number of λ windows constrained by the accuracy requirement.

Table 4.4 shows the results of running adaptive quadrature on 5 protein ligand systems, comparing the Total Task Execution Time and accuracy versus the non-adaptive case. The number of lambda windows are reduced on average by 32 %, hence reducing Total Task Execution Time by the same amount. The error on the adaptive results is also decreased, on average by 77 % (see fig. 4.4). More importantly, the error on all of the systems are reduced to below $0.2\,\mathrm{kcal\,mol^{-1}}$, which has recently been shown to be the upper bound of reproducibility across different simulation engines [61].

The Total Task Execution Time of the TYK2 L7–L8 system has increased for the adaptive run by 1 λ window, compared to the non-adaptive case. This is due to the

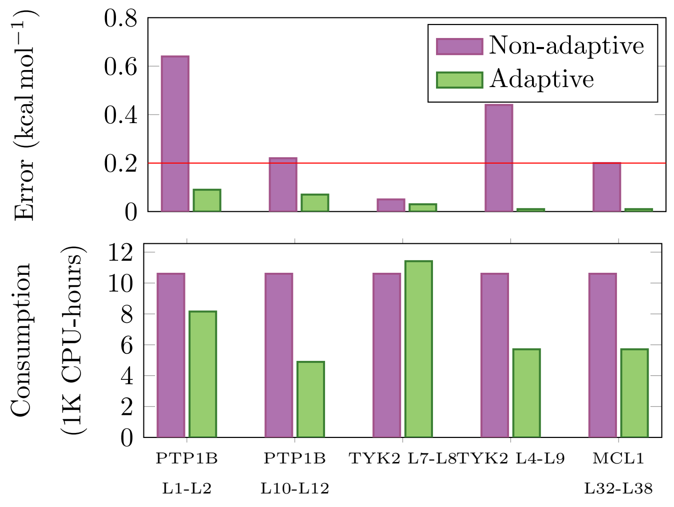


Figure 4.4: Quantifying the benefits of the adaptive quadrature simulations. (top) The error of the adaptive run is reduced for all 5 test systems, sometimes by a significant amount. It has been shown that reproducibility of free energy calculations can be achieved up to 0.2 kcal mol^{-1} [61]. The adaptive algorithm brings down the error of the nonadaptive simulations below this threshold, ensuring that results are also reproducible. (bottom) Resource consumption is reduced, except for one of the systems, where the low error threshold required more λ windows.

non-adaptive error being very low, and matching that same accuracy required the use of a large number of windows. Nonetheless due to the efficient placing of the windows, the accuracy of the free energy still increased by 40 %.

Fig. 4.5 compares the error on the adaptive and non-adaptive simulations as a time series plot. As fewer lambda windows are calculated the adaptive algorithm uses less resources. Remarkably, the error is drastically reduced as the windows are placed adaptively to capture the changes in function.

Adaptive quadrature is specific to alchemical free energy calculations. *Adaptive termination*, the second adaptive method implemented in HTBAC, offers dynamic termination for any simulation protocol that has as its aim the prediction of an observable value. The protocol monitors the convergence of the observable as the simulation progresses, and stops the execution when a criterion has been met. Non-adaptive protocols

Table 4.4: Comparing results of adaptive, non-adaptive and reference runs.

System	Ref ΔΔG (kcal mol^{-1})	Non-adaptive ΔΔG (kcal mol^{-1})	Adaptive ΔΔG (kcal mol^{-1})	No. of λ windows	Decrease in TTX	Increase in accuracy
PTP1B L1-L2	−58.51	−57.87(64)	−58.60(9)	10	23 %	86 %
PTP1B L10-L12	1.83	2.05(22)	1.94(7)	6	54 %	68 %
MCL1 L32-L38	2.13	2.33(20)	2.14(1)	7	46 %	95 %
TYK2 L4-L9	−28.69	−28.25(44)	−28.67(1)	7	46 %	98 %
TYK2 L7-L8	4.97	4.92(5)	5.00(3)	14	−8 %	40 %

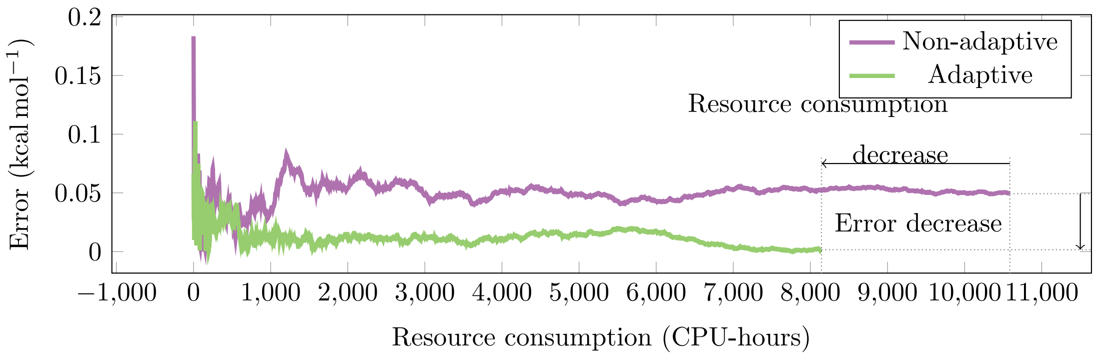


Figure 4.5: Plot of the error estimate as a function of the resource consumption, comparing the adaptive and nonadaptive simulations. The error estimate converges for both simulations but the window placement of the adaptive simulation considerably lowered the error.

usually have a predefined simulation time, set based on the assumption that the simulation will converge by that time. This means that in practical examples the simulation might have converged before the predefined simulation time.

In the original TIES protocol the production part of the simulation has to be run for 4 ns and the results are analyzed thereafter. This assumes that all systems need this simulation time for the results to converge. In reality, certain systems could converge faster, therefore one can terminate the simulation before the static 4 ns end. This would lead to faster time to insight and less compute resources consumed. Adaptive termination was implemented in HTBAC by having a checkpoint every $\tau = 0.5$ ns in the simulation. Fig. 4.6 shows how the observable for a specific simulation changes as a function of resource consumption. At every checkpoint the convergence is evaluated, and the simulation is indeed terminated earlier than the non-adaptive protocol would

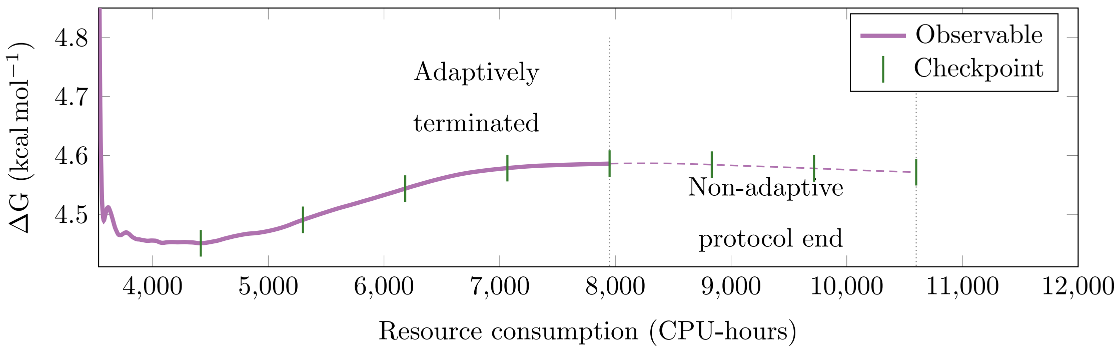


Figure 4.6: Plot of the free energy as a function of the resources consumed (hence simulation time). The adaptive termination algorithm checks the convergence of the observable every $\tau = 0.5\,\text{ns}$ and if the threshold ($0.01\,\text{kcal}\,\text{mol}^{-1}$) has been met, terminates the simulation.

Table 4.5: Simulation time of non-adaptive and adaptively terminated runs for a given convergence criterion

System	Non-adaptive	Adaptive	Decrease in TTX
PTP1B L10-L12	6.0ns	5.0ns	16.7%
TYK2 L4-L9	6.0ns	5.5ns	8.3%
TYK2 L7-L8	6.0ns	4.5ns	25.0%

suggest. Table 4.5 shows results that the adaptively terminated TIES protocol saves compute resources and reduces time to insight on average by 16 % for the physical systems tested.

Chapter 5
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. Ensemble-based binding affinity protocols have considerable predictive potential and enable reproducibility of results, deeming them to be ideal for computational drug campaigns. However as drug screening can cover millions of compounds and hundreds of millions of core-hours, it is important for ensemble-based binding affinity calculations to optimize the accuracy and precision of individual drug candidates. The ability to flexibly scale and adapt ensemble-based protocols to the drug candidate of interest is vital to produce reliable and accurate results on timescales which make it viable to influence real world decision making. However, the optimal protocol configuration for a given drug candidate is difficult to determine *a priori*, thus requiring runtime adaptations to workflow executions. In this thesis, we introduce HTBAC to enable ensemble-based approaches at scale on HPC, tailoring to computational drug campaigns.

Specifically, this thesis makes the following contributions: (1) discusses the landscape of ensemble-based simulations and their computational challenges (2) provides the design of a software tool (HTBAC) which addresses both challenges of scalability and adaptivity (3) characterizes the performance of HTBAC using a set of weak and strong scaling experiments (4) provide the adaptive execution capabilities to enable our users (primarily at UCL) to design and execute adaptivity experiments that improve scientific metrics, by increasing free energy accuracy at a fixed computational cost, or reducing the number of total simulations and thereby reducing computational load within an acceptable bounds of free energy errors.

In Chapter 4, we have characterized the scalability performance of HTBAC on NCSA

Blue Waters. We showed near-ideal weak and strong scaling behavior using two well-known binding affinity protocols (ESMACS and TIES), individually and together, to reach scales of 21,120 cores. This permits a time-to-solution that is essentially invariant of the size of candidate ligands, as well as the type and number of protocols concurrently employed. Furthermore, we validated binding free energies computed using HTBAC with both experimental and previously published computational results.

We also included a section on adaptivity experiments, devised by our users, in which they compared resource consumption and free energy accuracy using adaptive and non-adaptive TIES results. Using the adaptive quadrature algorithm, they showed improvements in $\Delta\Delta$G on average by 77% over the 5 physical systems tested. By reducing the λ windows on average by 32%, they reduced execution time by the same amount. The adaptive termination implementation of the TIES protocol saves compute resources and reduces time to solution on average by 16%. HTBAC's adaptivity capabilities allowed our users to develop new methodologies like the adaptive TIES protocol, which produced unprecedented scientific results. The adaptivity experiments validated the adaptive and scalable capabilities of HTBAC to enable users in their abilities to produce interesting research.

The use of software implementing well-defined abstractions like that of "building blocks", future proofs users of HTBAC to evolving hardware platforms, while providing immediate benefits of scale and support for a range of different applications. In this work we use the middleware provided by the RADICAL-Cybertools to execute applications of HTBAC. HTBAC represents an important advance towards the use of ensemble-based binding affinity calculations to the point where they can produce actionable results both in the clinic and industrial drug discovery.

The majority of our performance overheads are attributed to the the task launching delay arising from the use of RADICAL-Pilot and it increases as a function of the number of tasks simultaneously submitted for execution. RADICAL-Pilot is a new pilot system and is under constant development, as such more development is needed to improve the scheduling of highly concurrent tasks. This issue is now being addressed using scheduling decisions based on data awareness which we believe will mitigate this overhead even further.

The development of HTBAC has paved a platform to allow a significant increase in the size of studying drug candidates in computational drug campaigns. Much of the literature on MD-based binding affinity calculations is limited to a few tens of physical systems, usually of similar drugs bound to the same protein target. By facilitating the investigation of much larger sets of physical systems, HTBAC contributes to solve the grand challenge in drug design and precision medicine: understanding the influences on binding strength for hundreds or thousands of drug-protein variant combinations.

We believe that design decisions we have made, enable HTBAC to provide a high level of generality for ensemble-based approaches. HTBAC is designed to be a research vehicle for the domain scientists and to enable them to prototype and test ensemble-based approaches unavailable in other software frameworks. Design and modularity of the code significantly lower the barrier for implementation of novel scientific methodologies and reduce development time.

Chapter 6

Future Work

The most beneficial task in the context of this project would be developing a better understanding of different adaptivity uses cases, in order to improve the design and generality of HTBAC. It is crucial to identify scenarios and applications that span other ensemble-based approaches, besides computational drug campaigns. Also, it is important to provide a clear guidelines on how to maximize benefits arising from the usage of adaptive execution, less from an application perspective and more from an execution perspective. In addition to utilization experiments presented in Chapter 4, there are a number of scenarios which we have not covered. For example, we can develop protocols that use ensemble-based simulations, coupled with more complex analysis such as deep learning. In principle, the design of the HTBAC is invariant to the type of analysis task; however, we have not yet presented any experiments to validate this claim.

Another direction for future work is multi-cluster or multiple engine execution of HTBAC applications. We are seeing more applications surfacing that involve executing the analysis steps (machine learning models) on GPU clusters, while executing simulations in parallel on either GPU or non-GPU dominant clusters using multiple MD engines. Therefore, it is important to enable these scenarios to optimize the utility of all available computational resources. HTBAC can be improved in terms of its features and functionality by providing explicit support for simulations and analyses utilizing multiple heterogeneous HPC resources and execution engines.

References

[1] Steven M. Paul, Daniel S. Mytelka, Christopher T. Dunwiddie, Charles C. Persinger, Bernard H. Munos, Stacy R. Lindborg, and Aaron L. Schacht. How to improve R&amp;D productivity: the pharmaceutical industry's grand challenge. *Nature Reviews Drug Discovery*, 9(3):203–214, March 2010.

[2] David L. Mobley and Pavel V. Klimovich. Perspective: Alchemical free energy calculations for drug discovery. *The Journal of Chemical Physics*, 137(23):230901, 2012.

[3] Antonia S. J. S. Mey, Jordi Juárez Jiménez, and Julien Michel. Impact of domain knowledge on blinded predictions of binding energies by alchemical free energy calculations. *Journal of Computer-Aided Molecular Design*, Nov 2017.

[4] Jian Yin, Niel M. Henriksen, David R. Slochower, Michael R. Shirts, Michael W. Chiu, David L. Mobley, and Michael K. Gilson. Overview of the sampl5 host–guest challenge: Are we doing better? *Journal of Computer-Aided Molecular Design*, 31(1):1–19, Jan 2017.

[5] Lingle Wang, Yujie Wu, Yuqing Deng, Byungchan Kim, Levi Pierce, Goran Krilov, Dmitry Lupyan, Shaughnessy Robinson, Markus K. Dahlgren, Jeremy Greenwood, Donna L. Romero, Craig Masse, Jennifer L. Knight, Thomas Steinbrecher, Thijs Beuming, Wolfgang Damm, Ed Harder, Woody Sherman, Mark Brewer, Ron Wester, Mark Murcko, Leah Frye, Ramy Farid, Teng Lin, David L. Mobley, William L. Jorgensen, Bruce J. Berne, Richard A. Friesner, and Robert Abel. Accurate and Reliable Prediction of Relative Ligand Binding Potency in Prospective Drug Discovery by Way of a Modern Free-Energy Calculation Protocol and Force Field. *J. Am. Chem. Soc.*, 137(7):2695–2703, February 2015.

[6] Aaron Weis, Kambiz Katebzadeh, Pär Söderhjelm, Ingemar Nilsson, and Ulf Ryde. Ligand affinities predicted with the mm/pbsa method: dependence on the simulation method and the force field. *Journal of medicinal chemistry*, 49(22):6596–6606, 2006.

[7] A. P. Bhati, S. Wan, D. W. Wright, and P. V. Coveney. Rapid, accurate, precise and reliable relative free energy prediction using ensemble based thermodynamic integration. *J. Chem. Theory Comput.*, 13(1):210–222, 2017.

[8] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kalé, and K. Schulten. Scalable molecular dynamics with NAMD. *J. Comput. Chem.*, 26(16):1781–1802, 2005.

[9] T. P. Straatsma and J. A. McCammon. Multiconfiguration thermodynamic integration. *The Journal of Chemical Physics*, 95(2):1175–1188, July 1991.

[10] S. Wan, A. P. Bhati, S. J. Zasada, I. Wall, D. Green, P. Bamborough, and P. V. Coveney. Rapid and reliable binding affinity prediction of bromodomain inhibitors: a computational study. *J. Chem. Theory Comput.*, 13(2):784–795, 2017.

[11] Pavel V. Klimovich, Michael R. Shirts, and David L. Mobley. Guidelines for the analysis of free energy calculations. *Journal of Computer-Aided Molecular Design*, 29(5):397–411, May 2015.

[12] Levi N. Naden and Michael R. Shirts. Rapid Computation of Thermodynamic Properties over Multidimensional Nonbonded Parameter Spaces Using Adaptive Multistate Reweighting. *Journal of Chemical Theory and Computation*, 12(4):1806–1823, April 2016.

[13] Joseph W. Kaus, Levi T. Pierce, Ross C. Walker, and J. Andrew McCammon. Improving the Efficiency of Free Energy Calculations in the Amber Molecular Dynamics Package. *Journal of Chemical Theory and Computation*, 9(9):4131–4139, September 2013.

[14] Simona Soverini, Andreas Hochhaus, Franck E Nicolini, Franz Gruber, Thoralf Lange, Giuseppe Saglio, Fabrizio Pane, Martin C Müller, Thomas Ernst, Gianantonio Rosti, Kimmo Porkka, Michele Baccarani, Nicholas C P Cross, and Giovanni Martinelli. Bcr-abl kinase domain mutation analysis in chronic myeloid leukemia patients treated with tyrosine kinase inhibitors: recommendations from an expert panel on behalf of european leukemianet. *Blood*, 118:1208–1215, August 2011.

[15] Ahmet Zehir, Ryma Benayed, Ronak H Shah, Aijazuddin Syed, Sumit Middha, Hyunjae R Kim, Preethi Srinivasan, Jianjiong Gao, Debyani Chakravarty, Sean M Devlin, Matthew D Hellmann, David A Barron, Alison M Schram, Meera Hameed, Snjezana Dogan, Dara S Ross, Jaclyn F Hechtman, Deborah F DeLair, JinJuan Yao, Diana L Mandelker, Donavan T Cheng, Raghu Chandramohan, Abhinita S Mohanty, Ryan N Ptashkin, Gowtham Jayakumaran, Meera Prasad, Mustafa H Syed, Anoop Balakrishnan Rema, Zhen Y Liu, Khedoudja Nafa, Laetitia Borsu, Justyna Sadowska, Jacklyn Casanova, Ruben Bacares, Iwona J Kiecka, Anna Razumova, Julie B Son, Lisa Stewart, Tessara Baldi, Kerry A Mullaney, Hikmat Al-Ahmadie, Efsevia Vakiani, Adam A Abeshouse, Alexander V Penson, Philip Jonsson, Niedzica Camacho, Matthew T Chang, Helen H Won, Benjamin E Gross, Ritika Kundra, Zachary J Heins, Hsiao-Wei Chen, Sarah Phillips, Hongxin Zhang, Jiaojiao Wang, Angelica Ochoa, Jonathan Wills, Michael Eubank, Stacy B Thomas, Stuart M Gardos, Dalicia N Reales, Jesse Galle, Robert Durany, Roy Cambria, Wassim Abida, Andrea Cercek, Darren R Feldman, Mrinal M Gounder, A Ari Hakimi, James J Harding, Gopa Iyer, Yelena Y Janjigian, Emmet J Jordan, Ciara M Kelly, Maeve A Lowery, Luc G T Morris, Antonio M Omuro, Nitya Raj, Pedram Razavi, Alexander N Shoushtari, Neerav Shukla, Tara E Soumerai, Anna M Varghese, Rona Yaeger, Jonathan Coleman, Bernard Bochner, Gregory J Riely, Leonard B Saltz, Howard I Scher, Paul J Sabbatini, Mark E Robson, David S Klimstra, Barry S Taylor, Jose Baselga, Nikolaus Schultz, David M Hyman, Maria E Arcila, David B Solit, Marc Ladanyi, and Michael F Berger. Mutational landscape of metastatic cancer revealed from prospective clinical sequencing of 10,000 patients. *Nature medicine*, 23:703–713, June 2017.

[16] DB Longley and PG Johnston. Molecular mechanisms of drug resistance. *The Journal of pathology*, 205(2):275–292, 2005.

[17] U. S. Food and Drug Administration. Hematology/oncology (cancer) approvals & safety notifications, 2015. [Online; accessed 9-May-2015].

[18] Zheng Zhao, Hong Wu, Li Wang, Yi Liu, Stefan Knapp, Qingsong Liu, and Nathanael S Gray. Exploration of type ii binding mode: A privileged approach for kinase inhibitor focused drug discovery? *ACS Chem. Biol.*, 9(6):1230–1241, 2014.

[19] American Cancer Society. Cancer facts & figures 2015, 2015. [Online; accessed 7-May-2015].

[20] K. Marias, D. Dionysiou, V. Sakkalis, N. Graf, R. M. Bohle, P. V. Coveney, S. Wan, A. Folarin, P. Büchler, M. Reyes, G. Clapworthy, E. Liu, J. Sabczynski, T. Bily, A. Roniotis, M. Tsiknakis, E. Kolokotroni, S. Giatili, C. Veith, E. Messe, H. Stenzhorn, Yoo-Jin Kim, S. Zasada, A. N. Haidar, C. May, S. Bauer, T. Wang, Y. Zhao, M. Karasek, R. Grewer, A. Franz, and G. Stamatakos. Clinically driven design of multi-scale cancer models: the contracancrum project paradigm. *Interface Focus*, 1(3):450–461, 2011.

[21] P.M.A. Sloot, Peter V. Coveney, G. Ertaylan, V. Müller, C.A. Boucher, and M. Bubak. HIV decision support: from molecule to man. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 367(1898):2691–2703, 2009.

[22] Peter Kasson and Shantenu Jha. Adaptive ensemble simulations of biomolecules. *Current Opinions in Structural Biology*, 2018. `https://arxiv.org/pdf/1809.04804.pdf`.

[23] Vivek Balasubramanian, Travis Jensen, Matteo Turilli, Peter Kasson, Michael Shirts, and Shantenu Jha. Implementing adaptive ensemble biomolecular applications at scale. 2018. `https://arxiv.org/pdf/1804.04736.pdf`.

[24] M Karplus and J Kuriyan. Molecular dynamics and protein function. *Proc. Natl. Acad. Sci. U.S.A.*, 102:6679–6685, May 2005.

[25] S. Wan, B. Knapp, D. W. Wright, C. M. Deane, and P. V. Coveney. Rapid, precise, and reproducible prediction of peptide–MHC binding affinities from molecular dynamics that correlate well with experiment. *J. Chem. Theory Comput.*, 11(7):3346–3356, 2015.

[26] S. K. Sadiq, D. W. Wright, O. A. Kenway, and P. V. Coveney. Accurate ensemble molecular dynamics binding free energy ranking of multidrug-resistant HIV-1 proteases. *J. Chem. Inf. Model.*, 50(5):890–905, 2010.

[27] D. W. Wright, B. A. Hall, O. A. Kenway, S. Jha, and P. V. Coveney. Computing clinically relevant binding free energies of HIV-1 protease inhibitors. *J. Chem. Theory Comput.*, 10(3):1228–1241, 2014.

[28] S. Wan and P. V. Coveney. Rapid and accurate ranking of binding affinities of epidermal growth factor receptor sequences with selected lung cancer drugs. *J. R. Soc. Interface*, 8(61):1114–1127, 2011.

[29] S. Wan, A. P. Bhati, S. Skerratt, K. Omoto, V. Shanmugasundaram, S. K. Bagal, and P. V. Coveney. Evaluation and characterization of trk kinase inhibitors for the treatment of pain: Reliable binding affinity predictions from theory and computation. *J. Chem. Inf. Model*, 57(4):897–909, 2017.

[30] T. D. Bunney, S. Wan, N. Thiyagarajan, L. Sutto, S. V. Williams, P. Ashford, H. Koss, M. A. Knowles, F. L. Gervasio, P. V. Coveney, and M. Katan. The effect of mutations on drug sensitivity and kinase activity of fibroblast growth factor receptors: a combined experimental and theoretical study. *EBioMedicine*, 2(3):194–204, 2015.

[31] Gregory R. Bowman, Daniel L. Ensign, and Vijay S. Pande. Enhanced modeling via network theory: Adaptive sampling of markov state models. *Journal of Chemical Theory and Computation*, 6(3):787–794, 2010.

[32] David J. Earl and Michael W. Deem. Parallel tempering: Theory, applications, and new perspectives. *Phys. Chem. Chem. Phys.*, 7:3910–3916, 2005.

[33] Jozef Hritz and Chris Oostenbrink. Hamiltonian replica exchange molecular dynamics using soft-core interactions. *Journal of Chemical Physics*, 128(14):144121, 2008.

[34] Jaegil Kim, John E. Straub, and Tom Keyes. Replica exchange statistical temperature molecular dynamics algorithm. *The Journal of Physical Chemistry B*, 116(29):8646–8653, 2012. PMID: 22540354.

[35] A. P. Lyubartsev, A. A. Martsinovski, S. V. Shevkunov, and P. N. Vorontsov-Velyaminov. New approach to Monte Carlo calculation of the free energy: Method of expanded ensembles. *The Journal of Chemical Physics*, 96(3):1776–1783, February 1992.

[36] Anita de Ruiter, Stefan Boresch, and Chris Oostenbrink. Comparison of thermodynamic integration and bennett acceptance ratio for calculating relative protein-ligand binding free energies. *Journal of Computational Chemistry*, 34(12):1024–1034, 2013.

[37] Anita de Ruiter and Chris Oostenbrink. Extended thermodynamic integration: Efficient prediction of lambda derivatives at nonsimulated points. *Journal of Chemical Theory and Computation*, 12(9):4476–4486, 2016. PMID: 27494138.

[38] Gallivan Van Engelen. The gsoap toolkit for web services and peer-to-peer computing networks. *Cluster Computing and the Grid IEEE/ACM International Symposium*, 1(1):128–128, 2002.

[39] Matsuoka Tanaka. Ninf-g: A reference implementation of rpc-based programming middleware for grid computing. *Journal of Grid Computing*, 1(1):41–51, 2003.

[40] Boku Sato and Takahashi. Omnirpc: a grid rpc system for parallel programming in cluster and grid environment. *Cluster Computing and the Grid Proceedings CCGRID. 3rd IEEE/ACM International Symposium*, 1(1):206–113, 2003.

[41] Wil MP van der Aalst and Stefan Jablonski. Dealing with workflow change: identification of issues and solutions. *Computer systems science and engineering*, 15(5):267–276, 2000.

[42] Matteo Turilli, Yadu Nand Babuji, Andre Merzky, Ming Tai Ha, Michael Wilde, Daniel S. Katz, and Shantenu Jha. Evaluating distributed execution of workloads. *accepted IEEE eScience*, 2017. http://arxiv.org/pdf/1605.09513.

[43] Alessio Angius, Danila Oleynik, Sergey Panitkin, Matteo Turilli, Kaushik De, Alexei Klimentov, Sarp H Oral, Jack C Wells, and Shantenu Jha. Converging high-throughput and high-performance computing: A case study. *arXiv preprint arXiv:1704.00978*, 2017.

[44] Antons Treikalis, Andre Merzky, Haoyuan Chen, Tai-Sung Lee, Darrin M York, and Shantenu Jha. Repex: A flexible framework for scalable replica exchange molecular dynamics simulations. In *Parallel Processing (ICPP), 2016 45th International Conference on*, pages 628–637. IEEE, 2016.

[45] Vivekanandan Balasubramanian, Antons Treikalis, Ole Weidner, and Shantenu Jha. Ensemble toolkit: Scalable and flexible execution of ensembles of tasks. In *Parallel Processing (ICPP), 2016 45th International Conference on*, pages 458–463. IEEE, 2016.

[46] Vivekanandan Balasubramanian, Iain Bethune, Ardita Shkurti, Elena Breitmoser, Eugen Hruska, Cecilia Clementi, Charles Laughton, and Shantenu Jha. Extasy: Scalable and flexible coupling of md simulations and advanced sampling techniques. In *Proceedings of the 2016 IEEE 12th International Conference on e-Science*, pages 361–370. IEEE, 2016.

[47] I. Massova and P.A. Kollman. Computational alanine scanning to probe protein-protein interactions: A novel approach to evaluate binding free energies. *J. Am. Chem. Soc.*, 121(36):8133–8143, 1999.

[48] T. P. Straatsma and H. J. C. Berendsen. Free energy of ionic hydration: Analysis of a thermodynamic integration technique to evaluate free energy differences by molecular dynamics simulations. *The Journal of Chemical Physics*, 89(9):5876–5886, 1988.

[49] S. K. Sadiq, D. W. Wright, S. J. Watson, S. J. Zasada, I. Stoica, and P.V. Coveney. Automated Molecular Simulation Based Binding Affinity Calculator for Ligand-Bound HIV-1 Proteases. *J. Chem. Inf. Model.*, 48(9):1909–1919, 2008.

[50] Kresten Lindorff-Larsen, Stefano Piana, Kim Palmo, Paul Maragakis, John L. Klepeis, Ron O. Dror, and David E. Shaw. Improved side-chain torsion potentials for the Amber ff99SB protein force field. *Proteins: Structure, Function, and Bioinformatics*, 78(8):1950–1958, 2010.

[51] J. Wang, R. M. Wolf, J. W. Caldwell, P. A. Kollman, and D. A. Case. Development and testing of a general Amber force field. *J. Comput. Chem.*, 25(9):1157–1174, 2004.

[52] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, J. A. Montgomery, Jr., T. Vreven, K. N. Kudin, J. C. Burant, J. M. Millam, S. S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G. A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J. E. Knox, H. P. Hratchian, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, P. Y. Ayala, K. Morokuma, G. A. Voth, P. Salvador, J. J. Dannenberg, V. G. Zakrzewski, S. Dapprich, A. D. Daniels, M. C. Strain, O. Farkas, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. V. Ortiz, Q. Cui, A. G. Baboul, S. Clifford, J. Cioslowski, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, C. Gonzalez, and J. A. Pople. Gaussian 03, Revision C.02. Gaussian, Inc., Wallingford, CT, 2004.

[53] D. A. Case, T. E. Cheatham, T. Darden, H. Gohlke, R. Luo, K. M. Merz, A. Onufriev, C. Simmerling, B. Wang, and R. J. Woods. The Amber biomolecular simulation programs. *J. Comput. Chem.*, 26(16):1668–1688, 2005.

[54] Natalie H Theodoulou, Nicholas CO Tomkinson, Rab K Prinjha, and Philip G Humphreys. Clinical progress and pharmacology of small molecule bromodomain inhibitors. *Current Opinion in Chemical Biology*, 33(Supplement C):58 – 66, 2016. Chemical genetics and epigenetics * Molecular imaging.

[55] Paul Bamborough, Hawa Diallo, Jonathan D. Goodacre, Laurie Gordon, Antonia Lewis, Jonathan T. Seal, David M. Wilson, Michael D. Woodrow, and Chun-wa Chung. Fragment-based discovery of bromodomain inhibitors part 2: Optimization of phenylisoxazole sulfonamides. *Journal of Medicinal Chemistry*, 55(2):587–596, 2012. PMID: 22136469.

[56] Vivek Balasubramanian, Matteo Turilli, Weiming Hu, Matthieu Lefebvre, Wenjie Lei, Ryan T. Modrak, Guido Cervone, Jeroen Tromp, and Shantenu Jha. Harnessing the power of many: Extensible toolkit for scalable ensemble applications. In *2018 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2018, Vancouver, BC, Canada, May 21-25, 2018*, pages 536–545, 2018.

[57] Matteo Turilli, André Merzky, Vivek Balasubramanian, and Shantenu Jha. Building blocks for workflow system middleware. In *18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2018, Washington, DC, USA, May 1-4, 2018*, pages 348–349, 2018.

[58] Matteo Turilli, Mark Santcroos, and Shantenu Jha. A comprehensive perspective on pilot-jobs. *ACM Computing Surveys*, 2017.

[59] Jumana Dakka and et al. High-throughput binding affinity calculations at extreme scales. *accepted Computational Approaches for Cancer Workshop, SC'17*, 2017. `http://arxiv.org/abs/1712.09168`.

[60] Andre Merzky, Matteo Turilli, Manuel Maldonado, and Shantenu Jha. Design and performance characterization of radical-pilot on titan. *CoRR*, abs/1801.01843, 2018.

[61] Hannes H Loeffler, Stefano Bosisio, Guilherme Duarte Ramos Matos, Donghyuk Suh, Benoit Roux, David L Mobley, and Julien Michel. Reproducibility of free energy calculations across different molecular simulation software, 2018.