Parameterized Workload Adaptation for Fork-Join Tasks with Dynamic Workloads and Deadlines

Marion Sudvarg
Jeremy Buhler
Roger D. Chamberlain
Chris Gill
James Buckley
Wenlei Chen

Marion Sudvarg, Jeremy Buhler, Roger D. Chamberlain, Chris Gill, James Buckley, and Wenlei Chen, “Parameterized Workload Adaptation for Fork-Join Tasks with Dynamic Workloads and Deadlines,” in *Proc. of IEEE 29th International Conference on Embedded and Real-Time Computing Systems and Applications* (RTCSA), August 2023, pp. 232-242.

McKelvey School of Engineering
Washington University in St. Louis

Dept. of Physics
Washington University in St. Louis

School of Physics and Astronomy
University of Minnesota, Twin Cities

Parameterized Workload Adaptation for Fork-Join Tasks with Dynamic Workloads and Deadlines

Marion Sudvarg, Jeremy Buhler,
Roger D. Chamberlain, Chris Gill, Jim Buckley
Washington University in St. Louis
{msudvarg, jbuhler, roger, cdgill, buckley}@wustl.edu

Wenlei Chen
University of Minnesota, Twin Cities
chen6339@umn.edu

***Abstract*—Many real-time systems run in dynamic environments where exogenous factors inform task workloads and deadlines, which may not be known prior to job release. A job of a task that would otherwise miss its deadline may adapt to remain schedulable by executing in a degraded state that reduces its workload. We suggest that such a task should adjust parameters of its computation over multiple dimensions to maintain schedulability while minimizing loss of utility, which we discuss for highly parallel fork-join tasks executing on a fixed number of dedicated processors. We identify the parameterized degrees of freedom over which workload can be adjusted, then characterize the impact of workload reduction on response time and utility. From this, we generate a Pareto-optimal surface over which efficient search, interpolation, and extrapolation enable online selection of task parameters at time of job release.**

We apply this approach to the Advanced Particle-astrophysics Telescope, a planned mission to perform real-time gamma-ray burst (GRB) localization using SWaP-constrained embedded hardware aboard an orbiting platform. Due to GRBs' dynamic and uncertain nature, the workload and deadline may not be known prior to job release. Nonetheless, even for bright GRBs that may otherwise take longer than a second to localize on candidate embedded hardware, our approach often enables sub-degree accuracy while meeting a 33 ms imposed deadline.

I. Introduction

Many real-time systems execute in dynamic environments where exogenous factors inform task workloads and latency requirements, which therefore may not be known prior to job release. If a job's workload cannot be completed in time, it nonetheless might be able to adjust its computation to provide an imprecise result prior to the deadline: anytime workloads [1] stop executing when their budget is exhausted, providing the current state of their results, while others support discrete execution modes corresponding to varying degrees of precision that can be selected prior to execution [2], [3].

However, anytime or discrete semantics may not fully capture the dimensions over which a task's workload can adapt to meet its deadline. Some computations have multiple parameterized degrees of freedom that can be adjusted from their nominal values. These can be categorical (e.g., selecting from among a collection of algorithms) or numeric. Numeric parameters typically take discrete values (e.g., the number of iterations to refine a result), though at fine granularity, they can be approximated as a continuous state space (e.g., the proportion of input data selected from a large set for processing). If an instance of a task is not schedulable when run using its *desired* computational mode, its utilization may be reduced or *compressed* by adjusting these parameters to guarantee completion while minimally degrading result utility.

In this paper, we consider the problem of parameterized workload adaptation for highly parallel fork-join tasks with dynamic workloads and deadlines executing on a fixed number of dedicated cores. We compress the task's workload by adjusting its parameters so as to maximize the utility of its result within the available running time. For example, in simultaneous localization and mapping (SLAM) systems [4], result utility can be scored objectively according to the relative pose error (RPE) [5]; compression should therefore seek to minimize RPE within the constraints of schedulability.

Several challenges must be addressed in the face of dynamic workloads and deadlines. Characterizing the objective as a closed-form function over multiple parameterized dimensions may be difficult, and finding optimal values for each parameter that satisfy the problem's dynamic constraints may be inefficient for online compression. Furthermore, while parameters must be assigned to satisfy schedulability under worst-case assumptions, avoiding unnecessary worst-case pessimism (i.e., overcompression) remains a goal of this work.

Our solution is to quantify loss empirically for a large set of states (i.e., joint parameter settings), constructing a monotonically-decreasing hull of hyperplanes between these states. The set of all states is reduced to a Pareto-optimal surface by sorting candidate states in order of worst-case response times for a target platform and removing those for which a *greater* response time yields a *lower* utility. At job release, this surface can be efficiently searched to find a state satisfying the dynamic constraints imposed by the job-specific workload and deadline. Selected parameter values define a computational mode prior to execution. Despite conservative parameter selection to guarantee schedulability under worst-case execution times, some application semantics enable less pessimism. For example, an execution time budget may be assigned to anytime subtasks to allow additional execution if assigned work is completed early, and alternative approaches of slack reclamation (such as we describe in Sec. VIII) are sometimes possible.

We apply our techniques to the Advanced Particle-astrophysics Telescope (APT) [6], a planned orbital observatory that will detect and localize gamma-ray bursts (GRBs) in real time using onboard embedded hardware that is highly

The research presented in this paper was supported in part by NSF grants CSR-1814739, CNS-17653503, CNS-2229290 (CPS) and NASA grant 80NSSC21K1741 and was performed on behalf of the APT collaboration.

2325-1301/23/$31.00 ©2023 IEEE
DOI 10.1109/RTCSA58653.2023.00035

constrained in size, weight, and power (SWaP). Localization is a highly parallel fork-join task with a workload and deadline that depend on the unique characteristics of each GRB [7]. We demonstrate that our approach enables efficient online compression and slack reclamation, allowing for rapid and accurate localization of even bright transient GRBs that may provide only a short window of opportunity for observation.

II. Background and Related Work

Several approaches have been explored for adapting workloads under resource-constrained recurrent execution. Elastic scheduling models offer frameworks for dynamic adaptation of task utilizations to avoid system overload. The elastic model for implicit-deadline tasks on a uniprocessor [8], [9] compresses each task's utilization proportionally to its *elasticity*, a value indicating its relative adaptability. The model was extended to constrained-deadline tasks [10], [11], to multiprocessors [12], to federated scheduling of parallel real-time tasks for which periods [13] and workloads [14] are compressed over *continuous* ranges, and to tasks for which discrete sets of candidate utilizations may be accommodated [15]. Under federated scheduling, each parallel task executes on cores dedicated according to a sufficiency condition [16]; task utilizations are compressed so that the assigned cores fit within those available. While several of these models compress workloads, they do not describe how best to adapt computation under the applied constraints. The discretely elastic model considers specific modes of execution, but this does not capture the multiple degrees of freedom over which a task's workload may be compressed. Recent work [17] suggests using subtask-specific objective functions but is limited to end-to-end sequential execution and does not consider how compressing a subtask may affect its successors.

In [18], the authors survey protocols for switching between execution modes without missing deadlines. An adaptive framework in [19] degrades task execution according to "service levels," allowing for a user-defined notion of quality of outcome. A similar approach was presented for parallel LiDAR object detection [3] and applied to the PointPillars [20] encoder, aiming to maximize utility while maintaining schedulability guarantees by profiling execution times and assigning accuracies to discrete states. Such approaches may allow for a more objective measure of performance than traditional elastic scheduling, which compresses task utilizations proportionally to their elasticity. However, they are limited in practice to a small set of discrete states. We propose to allow adaptation according to a user-defined objective function over multiple degrees of freedom involving continuous or discrete numeric values and categorical variables.

Recent work on dynamic deadline-driven execution [21] presents a novel adaptation scheme for tasks with environment-dependent deadlines and execution times. This data-driven execution model provides handlers for deadline misses, allowing downstream components in the computational pipeline to adapt in response. The authors argue that periodic execution models, which use conservative WCET estimates, fail "to maximize the runtime-accuracy trade-off due to the large skew between the mean and maximum runtime," which typically "leaves plenty of slack." In Sec. IX, we demonstrate an approach that reclaims slack to provide higher utility while still using conservative WCET estimates to prevent deadline misses.

Our work focuses on highly parallel fork-join tasks, for which it is straightforward to characterize a closed-form worst-case response time under a nearly optimal schedule as a function of its compressible parameters. Many real-world applications [22], [23], including APT's GRB localization task [7], [24], [25], can be described as such.

III. Problem Statement

We consider recurrent, constrained-deadline, highly parallel fork-join tasks. Each released *job* is characterized by workload C, representing its worst-case execution time on a single processor core; and relative deadline D, the interval after its release by which it must complete execution. The task's workload and deadline may be dynamic but are fixed for a given instance at job release. Such a task τ can be decomposed into a sequence of subtasks $\{\tau_i\}$ with workloads C_i, where each subtask is either *sequential* (s) or *parallel* (p), as is illustrated in Fig. 1 in Sec. V. We assume the task executes on a fixed number of dedicated cores n; as these are highly parallel tasks for which parallel subtasks have workloads that can be distributed evenly across processors, the worst-case response time $\mathcal{R}$ can be expressed as

$$\mathcal{R} = \sum_{\tau_i \text{ is } s} C_i + \sum_{\tau_i \text{ is } p} \frac{C_i}{n} \tag{1}$$

Every prior instance of a constrained-deadline task must complete before activation of its next job. This implies that the task is schedulable if and only if $\mathcal{R} \leq D$ for each instance.

In this paper, we address the scenario where a job is released with a workload and deadline for which it is not schedulable. We consider computationally elastic tasks having multiple execution states associated with a set of application-specific parameters $\{a_j\}$ over which workload can be adjusted according to one of the following semantics, allowing response time to be expressed as a monotone non-decreasing function $\mathcal{R}(\{a_j\})$:

1) The workload of one or more subtasks may be a function $C_i(\{a_j\})$ of discrete or continuous numeric parameters; these must be monotone non-decreasing in each parameter. Sec. VI provides examples of subtasks having execution times linear or quadratic in a continuous numeric parameter representing the amount of input data to process.
2) A discrete numeric parameter (e.g., a number of iterations) may change the sequence of subtasks. In this work, we consider the case where such a parameter defines the number of identical copies of a sequence of subtasks.
3) A categorical parameter may change the computational mode (e.g., the algorithm used) of one or more subtasks. In this case, each mode may impose its own workload as

a function $C_i(\{a_j\})$ of the other parameters. To match the semantics of (1), we assign numeric values to each category, with the requirement that a larger numeric value is assigned to a mode with a greater workload.

Each parameter a_j is constrained by some maximum value $a_j^{\max}$. These values may be either constants or monotone non-decreasing functions of another parameter. The *uncompressed* workload is defined as that associated with each parameter taking its maximum value. Formally, a task with a dynamic workload is one for which some values $a_j^{\max}$ are unknown prior to job release. If a job is not schedulable, its workload is *compressed* by selecting values $\{a_j\}$ such that its response time does not exceed its deadline. Compression should attempt to maximize the utility of the system by minimizing some application-specific loss function $\mathcal{L}(\{a_j\})$ of these parameters. We consider applications for which $\mathcal{L}$ is monotone non-increasing with each parameter, i.e., doing *more* work yields a better result. We formulate our problem as follows:

$$\min_{\{a_j\}} \quad \mathcal{L}(\{a_j\}) \tag{2a}$$

$$\text{s.t.} \quad \mathcal{R}(\{a_j\}) \leq D \tag{2b}$$

$$\forall_j,\ a_j^{min} \leq a_j \leq a_j^{max} \tag{2c}$$

Here, a_j^{min} constrains the parameter to some minimum value and is assumed to be known a priori.

The problem, then, is to identify the parameters over which a task's workload may be compressed; to characterize their impact on the task's response time and the utility of its result; and finally to use this information to solve optimization problem (2) efficiently online to adjust a released job's computation to adapt to overload.

IV. Solution Overview

In this section we provide a solution overview for highly parallel fork-join tasks executing on a fixed number of dedicated cores. In subsequent sections we then illustrate and evaluate our approach in the context of a real-world application.

Offline Steps. The following steps are performed first offline, enabling an efficient online solution search.

① Identify Parameters. Parameters may be identified offline by inspection of the application and should match the semantics listed in Sec. III. The parameters for adapting APT's GRB localization task are described in Sec. VI.

② Characterize an Objective Function. Utility loss for an application can be quantified empirically for a large set of input state combinations. For each parameter a_j, a number b_j of values within the state space should be considered. The complete Cartesian product of these values should be tested, except where some parameter is constrained by another. Smaller values of b_j reduce the number of samples for efficiency of offline analysis, but denser sampling may allow for more accurate characterization of the objective (and the input space may still be reduced in ④ for use online). The selection is left up to the application designer, though for categorical parameters, each possible value should be considered. In Sec. VI, we identify 4 parameters and test 2657 input states for GRB localization.

For x numeric and y categorical parameters, the objective will be a function of $x + y$ dimensions, characterized as $\prod_y b_j$ x-dimensional manifolds. Fitting a closed-form function to the losses observed at the sampled states may be difficult and error-prone. Instead, our approach allows the loss function to be represented as a monotonically decreasing hull formed by hyperplanes connecting the space of observed states. Construction from a Pareto-optimal subset of states is described in ④.

③ Quantify Response Time. The task's worst-case response time can be quantified by decomposing it into constituent subtasks, then profiling subtask execution times individually or in groups that share dependence on common parameters. Constituent execution time functions $C_i(\{a_j\})$ can thus be characterized for those parameters satisfying semantic (1) from Sec. III. For those parameters a_j satisfying semantic (2), some subset of subtasks is duplicated by the parameter value, equivalent to scaling the workload of the individual subtasks by a_j. This can be incorporated into the expression $C_i(\{a_j\})$. From this, Eqn. 1 can be used to compute response times as functions of execution times, allowing easy adjustments of core assignments for the application on a given platform. For categorical parameters, the response-time functions for the Cartesian product of their values must be identified, resulting in up to $\prod_y b_j$ functions of the form

$$\mathcal{R}(\{a_j\}) = \sum_{\tau_i \text{ is } s} C_i(\{a_j\}) + \sum_{\tau_i \text{ is } p} \frac{C_i(\{a_j\})}{n}. \tag{3}$$

While the set of functions grows exponentially in the number of categorical parameters, realistically this can be represented more compactly. Such parameters typically represent the selection between a handful of available computational modes or algorithms to apply to a phase of the application. These are selected by the application designer and so may be as small in number as desired. For example, in Sec. VI, a single categorical parameter selects one of two algorithms for an initial approximation of a gamma-ray burst's direction; this selection only affects the response time of the approximation stage subtasks. As the results in Sec. IX demonstrate, a single such parameter is sufficient for our target application.

④ Generate Pareto-Optimal Surface. Candidate states are sorted by response time, after which any state with a higher loss than the previous state (i.e., a higher response time results in a worse outcome) is discarded. As noted in Sec. VIII, for APT, this procedure yields fewer than 100 candidate states for each platform we tested. From these, we construct hyperplanes connecting adjacent states for interpolation. For each candidate state s, we find the points from the *original* set of states having the next larger value of each parameter respectively with lower error,[1] holding constant the other parameters in s. These hyperplanes can be extended for extrapolation to parameter values beyond the ranges used to infer the surface.

[1]Due to the often stochastic nature of characterizing loss, some adjacent states may not have a lower objective value for a larger parameter value.

Online Steps. To implement online task compression, we modify the task to include an initial sequential subtask that calculates its response time according to the revealed constraints on workload parameters at time of release. In an overload scenario, the subtask should then solve the optimization problem (2) and apply the resulting parameters. Realizing computational mode changes is application-specific, but we outline an OpenMP-based approach for GRB localization in Sec. VII. As this subtask adds to task workload, it must remain efficient and be accounted for in the response time.

⑤ **Check for Overload.** When a job of a dynamic task arrives, the initial subtask must determine if the job will complete in time. To do so, it must calculate response time based on the parameter constraints revealed. We assume that each function $C_i(\{a_j\})$ can be computed in time linear in the number of numeric parameters x, so for a given input state, response time can be calculated in time $\mathcal{O}(x_d m)$ for m subtasks and x_d numeric parameters with dynamic constraints.

⑥ **Online Solution Search.** If a job's response time exceeds its deadline, the Pareto-optimal surface can be searched for a set of parameters that satisfy schedulability. Binary search over the sorted set of candidates finds the state s with the greatest response time not exceeding the deadline, from which a Pareto-optimal solution is then obtained by interpolation or extrapolation. This can be performed efficiently by considering each parameter in s connected to an adjacent state in ④, with the other parameters held constant, solving in constant time for the value yielding a response time equal to the deadline. The best such value obtained (i.e., the one corresponding to the state with the lowest objective function value) is chosen. This takes total time linear in the number of numeric parameters.

In the case that the state s has values that exceed the dynamic parameter constraints imposed on the job, iterative search down from s can be used to find the best state s' for which all parameters are within the constraints. However, parameter extrapolation from this state is not guaranteed to find a Pareto-optimal set of parameters, though the values will have a higher expected utility than s'. This is not a problem for our target application, as its dynamic constraints (described in Sec. VI) are defined by the amount of input data available, and our real-world test cases (described in Sec. IX) all provide sufficient input data. As such, exploration of alternative approaches (such as storing multiple surfaces, or falling back to iterative search over the complete set of candidate states generated in ③) are deferred to future work.

⑦ **Adapt Task Execution.** For subtasks with discrete modes, execution should proceed according to the state defined by the input parameters, but this may result in overcompression as worst-case response times are often pessimistic. For collections of subtasks with anytime workloads, workload compression can instead be applied by calculating WCETs corresponding to the given input parameters. This portion of the task may then be allowed to proceed until the compressed WCET or response-time limit has been reached, whereupon it is stopped and the current result is used. Some applications may provide other opportunities to reduce pessimistic overcompression via slack reclamation; we describe one such approach for GRB localization in Sec. VIII.

V. Target Application: GRB Localization

The Advanced Particle-astrophysics Telescope (APT) [6] is a planned space-based observatory that will support multi-wavelength and multi-messenger astrophysics [26]–[28] by rapidly detecting gamma-ray bursts (GRBs) and directing follow-up instruments to observe GRBs across broad ranges of wavelengths and emission modalities. APT is designed for a nearly full-sky field of view, but many follow-up instruments have narrow apertures (often $<1^\circ$) and so must point almost directly at the GRB source. APT will do onboard detection and localization of GRBs in real-time, enabling prompt communication of precise source directions to those instruments.

We consider reconstruction and localization of Compton-regime GRBs using techniques presented in [24], [25]. Thousands to millions of gamma-ray photons enter APT's detector from a single burst. Each may Compton-scatter one or more times before being photoabsorbed; each interaction is referred to as a *hit*, and a single photon's hits are collectively referred to as an *event*. Event reconstruction, followed by localization of the GRB by combining multiple reconstructed events, both execute on a fixed number of cores in SWaP-constrained hardware flying aboard the orbital platform.

The GRB localization task forms the highly parallel fork-join computation illustrated in Fig. 1. The task can be decomposed into a sequence of subtasks that collectively form the three stages detailed in Sec. VI. Each instance of this task, corresponding to the detection and localization of a unique GRB, is highly variable in its workload and deadline. The workload depends on the number of detected events, which itself is a function of the burst's spectral-energy distribution, angle with respect to the detector, and fluence (a measure proportional to the number of incident gamma rays) [7]. The deadline may depend on the burst duration, which can range from around 10 ms to 20 minutes in the Compton regime [29]–[32]. It also may be informed by the communication latency and slewing speeds of available follow-up instruments. Speed-of-light delays to ground-based devices impose an extra $\approx$5 s of latency, but APT could be instead be coupled with an onboard optical telescope (similarly to Swift's UVOT [33]). For highly transient bursts, the window of opportunity for follow-up observations may be very brief, though the timescale of prompt emissions in secondary modalities is still an open question in astrophysics. The localization task therefore must

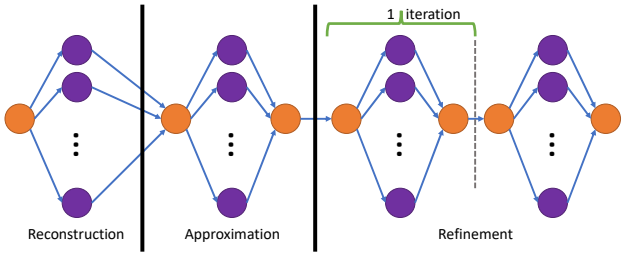


Fig. 1: APT's highly parallel fork-join localization task.

Param	Stage	Description	Constraint
n_r	Reconstruction	Events to reconstruct	$30 \leq n_r \leq n_e$
α	Approximation	Approximation technique	$\alpha \in \{\textbf{FibSpiral}, \textbf{ApproxCircles}\}$
n_s	Approximation	Number of annuli to sample for joint log-likelihood	$\max\{10, n_a\} \leq n_s \leq \min\{1000, n_a\}$
x	Refinement	Refinement iterations	$x \in \{0 \ldots 20\}$

TABLE I: Compressible parameters for APT localization task.

adaptable to guarantee completion before a deadline that may not be known a priori, even for highly transient bursts generating large volumes of data.

To characterize system performance, we simulate the instrument's response to several GRBs with the APTSoft package [34] that uses the Geant4 simulator [35] to generate independent gamma rays from a simulated source and track their physical interactions in the detector, then models the response of the front-end electronics. We generate sets of 10^6 gamma rays using two spectral-energy distributions characteristic of short GRBs [36]. We use two Band [37] functions with parameters α=−0.5, E_{peak}=490 keV, $\beta \in \{-3.2, -2.1\}$ to capture a range of spectral profiles. Spectral energies are in $[100\,\text{keV} - 30\,\text{MeV}]$ to match the Compton regime of the Fermi Gamma-ray Burst Monitor (GBM) sensitivity [38], data from which the distributions presented in [36] were obtained. For each spectrum, we first generate a normally-incident set, and then the sets described by the Cartesian product of $\{30°, 60°\}$ polar angles and $\{0°,\ 45°\}$ azimuth angles. This gives a total of 10 synthetic GRBs across 5 incident angles and 2 spectral energy distributions, which we use to characterize the pipeline's localization accuracy and worst-case execution times.

VI. Parameters and Loss Function

Workloads in each of APT's three pipeline stages may be compressed to fit a dynamic deadline known only when each job is released. We aim to minimize an objective function informed by the angular error in the predicted GRB source direction, while still guaranteeing that the deadline is met. The associated compressible parameters are outlined in Table I.

Stage 1: Event Reconstruction. For each event, we use the tree search algorithm from [24] to infer the temporal order of the first two hits, constraining the gamma ray's source vector to an annulus on the unit sphere with thickness determined by uncertainties in detector spatial and energy measurements [39]. In simulation, events with more than 6 hits are extremely uncommon ($<0.01\%$), so we exclude these from reconstruction to bound the tree traversal. Physically impossible reconstructions are dropped, with the remaining n_a annuli passed to localization. Reconstruction can degrade by dropping events: for n_e reconstructable events, we can select $n_r \leq n_e$ events to actually reconstruct. As n_e is typically on the order of several thousand or more, we approximate n_r as a continuous numeric variable. Reconstruction is an anytime workload: the stage can stop at any point.

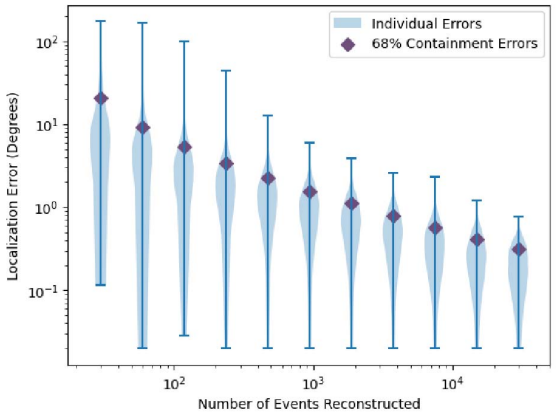


Fig. 2: Impact of n_r on localization error. Note that axes are logarithmic.

To characterize the impact of compressing n_r, we iterated over a geometric progression of 11 values from 30 to 30 000, using uncompressed values for all other input parameters. For each value of n_r, we generated 10 000 inputs to the pipeline by randomly sampling 1000 subsets of reconstructable events from each of the 10 simulated GRBs. Fig. 2 plots the discrepancy in degrees between the inferred and true source direction against the number of events reconstructed, with the vertical bars enclosing the extent of the distribution. Because of the high variance in localization error for a given value of n_r, rather than using expected error as the objective, we instead use 68% containment (representing the 68^{th} percentile error, a commonly used metric for GRB localization accuracy [34], [40]). These values are shown in Fig. 2, which illustrates a roughly log-log linear dependence on n_r.

Stage 2: Initial Source Approximation. We use multilateration over reconstructed annuli to infer the GRB's source direction. This involves an initial rough approximation that is then iteratively refined in Stage 3. We consider two approximation techniques (α) both of which execute over a subset $n_s \leq n_a$ of the input annuli. Our prior work [24], [25] fixed n_s=1000 and used only the first approximation technique (**ApproxCircles**). It uniformly distributes 720 points around each of 20 circles selected at random from n_s, finds the point from each with the greatest joint log-likelihood over all n_s annuli, then uses a weighted mean to approximate the source vector. The second technique (**FibSpiral**) is new to this work. It generates 100 points almost uniformly over the surface of the unit sphere with a Fibonacci spiral. For each point, it finds the joint log-likelihood over all n_s annuli, then approximates the source vector as a weighted mean over the top 10. Approximation requires both α and n_s to be specified prior to computation.

While **FibSpiral** is much faster (requiring only $100 \cdot n_s$ log-likelihood computations, versus $14\,000 \cdot n_s$ for **ApproxCircles**), it has less fidelity in its estimate for equal values of n_s. In this work, we constrain n_s to the range $[10, 1000]$, which with the choice of α approximates two continuous state spaces that are non-overlapping in execution time but may overlap in result accuracy, as illustrated in Fig. 3. Measured 68% con-

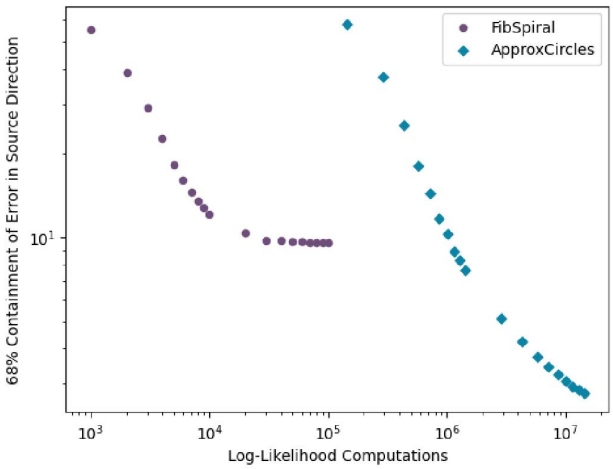


Fig. 3: Comparison of approximation techniques.

tainments for the approximated source error (degrees) without refinement are plotted against the number of log-likelihood computations required by values of n_s for each technique. For this plot, no subsequent refinement is performed, and n_r is fixed at 30 000. 68% containments for each n_s were obtained from 1000 trials over each simulated GRB.

Stage 3: Iterative Source Refinement. The approximation result is subsequently refined using a modified version of the iterative linear least-squares approach in [24], [25] over all reconstructed data. Refinement executes for x iterations (or until convergence). Whereas our prior work fixed x=20, now we allow the task to adapt by compressing x to a discrete numeric value in the range $\{0\ldots20\}$. Iterative refinement can be terminated at any time, with the result of the last completed iteration (or the initial approximation, if no iterations completed) used as the estimated source direction of the GRB.

Iterative refinement is highly dependent on the quality of the initial source estimate provided by approximation, as illustrated in Fig. 4. Each value of (n_s, x) is plotted against the 68% containment of localization error (degrees) over 1000 trials from our 10 synthetic GRBs with n_r=1893 (the smallest

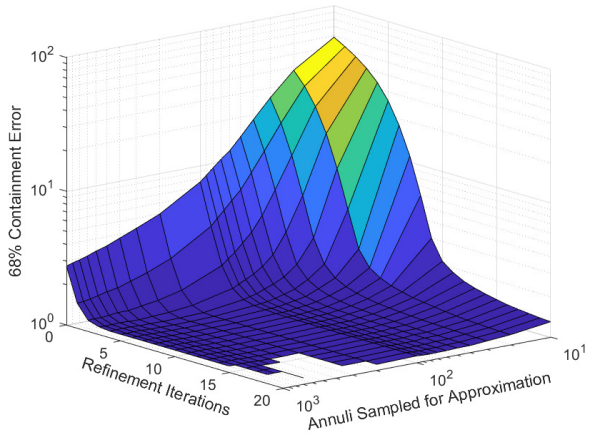


Fig. 4: Impact of approximation on iterative refinement.

value in our geometric progression for which n_s can reach 1000) and using the **ApproxCircles** technique. With fewer annuli sampled for approximation, more refinement iterations are necessary to converge on an accurate result. With more refinement iterations, the impact of a poor initial estimate is reduced.

VII. Response Times

Each of APT's three stages of CPU computation (illustrated in Fig. 1) has an initialization subtask preceding a parallel subtask, with the work evenly split across cores using OpenMP. We measure execution times for each subtask, then use these to calculate response times for each stage as functions of the tunable parameters.

Reconstruction processes events independently, with its workload linear in n_r after a constant-time initialization. To characterize response time, we fit a linear function on the three hardware platforms listed in Table II, all with CPU throttling disabled, running the pipeline at the highest real-time priority. These platforms span a range of performance in available SWaP-constrained hardware while remaining tolerant to harsh environmental conditions. Though not rad-hardened, they are candidates to fly aboard a scheduled Antarctic high-altitude balloon demonstration mission, and our group has flown the Atom-based platform on previous such missions.

On each platform, we profile the reconstruction stage for values of n_r from 3000 to 27 000 in steps of 3000. For each value, we collect 20 response times from each of our 10 simulated GRBs. To better capture the worst case, we collect 200 times from each GRB for n_r=30 000. We fit a linear function over the maximum times for each n_r, then offset by the greatest positive residual to guarantee the function upper-bounds all 3200 observed response times. Measured worst-case times and characteristic functions are illustrated in Fig. 5. Note that on the Atom, the extra samples for n_r=30 000 produced a slight outlier in measured WCET, indicating that the platform's timing is slightly less stable than the RPi3 or RPi4. Nonetheless, with our offsetting technique we were still able to meet all deadlines considered in Sec. IX.

Approximation initializes with results from reconstruction then samples annuli at random, with which it computes joint log-likelihoods for each of its candidate source directions. Aggregation and sampling are performed by a single subtask

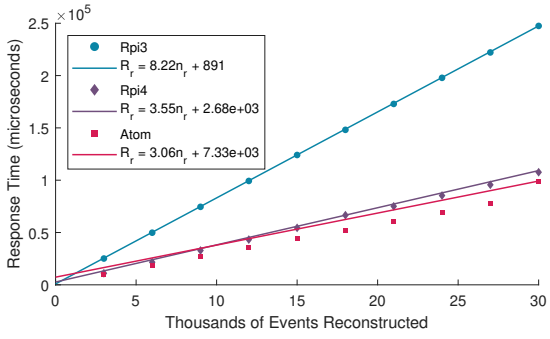
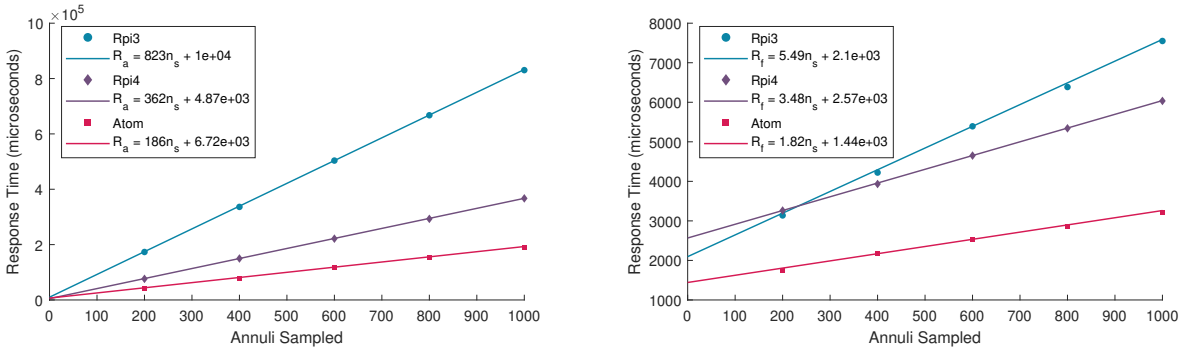


Fig. 5: Reconstruction stage worst-case response times.

Platform	Abbr.	CPU	Cores	Freq.	RAM	Linux Kernel
Raspberry Pi 3 Model B+	RPi3	Cortex-A53 (ARMv8)	4	700MHz*	1GB	5.15.61
Raspberry Pi 4 Model B	RPi4	Cortex-A72 (ARMv8)	4	600MHz*	4GB	5.15.61
WINSYSTEMS EBC-C413	Atom	Intel Atom E3845	4	1.92GHz	8GB	5.15.0

TABLE II: Hardware platforms evaluated. *While the Raspberry Pi models tested support higher CPU clock speeds, we use the lower frequencies recommended in [41], [42] to prevent throttling and instability.



(a) **ApproxCircles** (b) **FibSpiral**

Fig. 6: Approximation stage worst-case response times.

that precedes an independent subtask for each candidate source direction (100 for **FibSpiral** and 14 400 for **ApproxCircles**). A final subtask aggregates the results to find an average source vector, weighted by the joint log-likelihoods. Execution times for each subtask scale linearly with the number of sampled annuli n_s. We characterize the worst-case response time separately for each technique α. We profile both approximation techniques for $n_s \in \{200, 400, 600, 800, 1000\}$. For each value, we collect 20 response times from each of 10 simulated GRBs. Similar to reconstruction profiling, we fit a linear function over the maximum times for each n_s, then shift vertically to guarantee an upper bound over all observed response times. The measured worst-case times and corresponding functions are illustrated in Figures 6(a) and 6(b). Notice that the vertical axis scale for **ApproxCircles** is 2 orders of magnitude greater than for **FibSpiral**, commensurate with the number of log-likelihood computations required by each technique.

Finally, **Refinement** is iterative; each iteration has a sequential initialization subtask followed by a parallel subtask to process and filter each annulus. A final sequential subtask in each iteration constructs and solves a constant-time quadratic eigenvalue problem, for which forming the matrix has cost quadratic in the number of reconstructed annuli [6]. Each iteration, then, has a worst-case response time quadratic in n_r; this is multiplied by x to produce the response time of the stage. We fit this function on our three candidate devices, profiling each iteration of refinement from the same set of runs measured for reconstruction. Results are illustrated in Fig. 7.

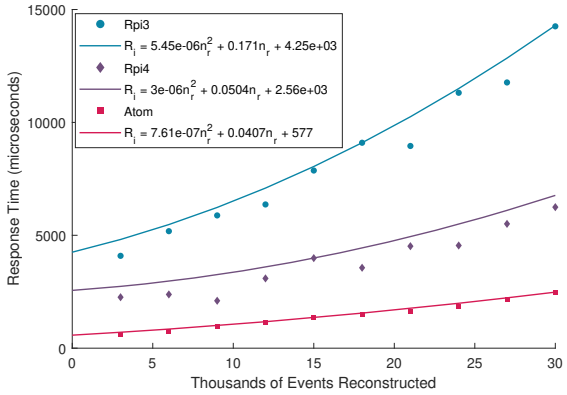


Fig. 7: Refinement stage worst-case response times.

VIII. Implementation

In this section, we discuss our implementation of workload adaptation for APT's GRB localization task.

Pareto-Optimal Surface. We quantified response times for 2657 input parameter states per the functions identified for each stage in Sec. VII. After generating a Pareto-optimal set of candidates (Sec. IV), only 81 states remained for the RPi3, 84 for the RPi4, and 83 for the Atom. We characterized localization error for all values of α and x for each tested value of n_r and n_s, removing the option of interpolation over these discrete states. For each candidate state s, we reduced the hyperplanes connecting adjacent values of n_r and n_s to log-linear functions of error in each of these parameters. As the maximum value of n_r is a dynamic constraint, we also constructed log-linear functions from the points for which n_r=30 000 (the largest value tested) by extrapolation from the state with the next smaller value of n_r having a higher measured error. Each candidate state and its log-linear function parameters are stored in a lookup table to use online.

Determining Parameter Values. When a job arrives, the localization task checks the worst-case response time for the number of reconstructable events: if it exceeds the deadline, parameter values are selected accordingly (Sec. IV).

Adapting to Overload. The number of events reconstructed affects the response time of the downstream refinement stage, so we do not treat reconstruction as an anytime workload. Instead, once parameter values are selected, global variables are set prior to computation to restrict the number of events reconstructed, the number of annuli sampled for refinement, and

the number of refinement iterations to perform. The software implements **FibSpiral** and **ApproxCircles** as C++ subclasses of a common **Approximation** class, allowing dynamic object construction according to the chosen value of α.

Slack Reclamation. When parameter values for an execution mode guarantee completion before the deadline in the worst case, pessimism in WCET estimates may result in overcompression. Nonetheless, some tasks provide opportunities for slack reclamation after computation completes. APT's final iterative source refinement is an anytime workload, so slack could be reclaimed naïvely by allowing it to continue iterating until the deadline. However, it might still complete early if refinement converges, and this does not consider that earlier stages of the pipeline may also have been compressed. Instead, we implement a version of slack reclamation that determines, given the remaining slack time (less its own overhead), how many additional events can be reconstructed with another refinement iteration run over the resulting larger set of annuli. For efficiency, rather than allowing OpenMP to split events among threads prior to reconstruction, an idle thread retrieves an event using an atomic fetch-and-increment of an index that tracks the next available event. Once this index reaches n_r, reconstruction halts, but is resumed when slack reclamation increases the limit. Reclamation continues in a loop until there is insufficient slack time remaining, as illustrated in Fig. 8. At this point, additional iterations of refinement can still be run until the deadline (or until convergence).

IX. Evaluation

To evaluate our proposed workload compression approach, we first measured the overheads associated with each optional extension of the APT GRB localization application (interpolation/extrapolation and slack reclamation) to appropriately account for them. We next compared these extensions when running the pipeline against our synthetic GRBs to decide which are expected to further improve results against real-world data. Finally, we evaluated our approach in the context of historical GRBs to demonstrate that our approach extends from training to real-world tests, enabling localization of bright GRBs even under highly constrained deadlines.

A. Overheads

To evaluate our proposed workload compression approach, we applied it to the APT localization task. We began with profiling the overhead of searching online for a Pareto-optimal set of compressed input parameters (described in Sec. IV) and of computing the inputs to slack reclamation (Sec. VIII). We ran the localization task with both compression and slack reclamation over the synthetic GRBs described in Sec. V. For each GRB, we tested numbers of input gamma-ray photons over a geometric progression of 9 values from 30 to 10^6. Geometric progressions of 9 deadline values were selected separately for each hardware platform to guarantee that the shortest deadline would be between the response times of the first two candidate states, and that the longest deadline would be greater than the response time of the last candidate state.

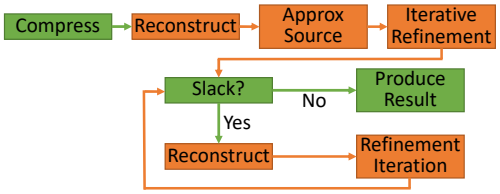


Fig. 8: Localization pipeline with compression and slack reclamation.

Fig. 9 illustrates the overheads of the 810 profiled runs of online compression for each tested hardware platform, with the vertical bars enclosing the distribution. The overhead remained under 220 μs on the RPi3, under 180 μs on the RPi4, and under 60 μs on the Atom, demonstrating the efficiency of our online compression technique. We adjusted the response time functions for each of our platforms accordingly.

Overheads of slack reclamation were captured by profiling the elapsed time of the first successful attempt to reclaim slack for each run. Those runs for which slack could not be reclaimed were ignored, as the overhead may be lower in these cases. This produced 538 samples for the RPi3, 555 for the RPi4, and 573 for the Atom; these are also illustrated in Fig. 9. Despite the equivalent program logic, the RPi3 had significantly higher overhead (notice the difference in vertical-axis units): its overhead reached 97.5 μs, whereas it remained under 1.9 μs on both the RPi4 and Atom.

B. Evaluation on Synthetic GRBs

To characterize the expected performance of our approach, we evaluated three versions when applied to APT's localization task. The first, **Pareto**, finds the best state from the Pareto-optimal set of candidates with a response time that does not exceed the deadline and for which $n_r \leq n_e$ according to the procedure in Sec. IV. The second, **IntExt**, additionally interpolates or extrapolates from that state. The third, **Reclaim**, performs compression equivalently to **IntExt** while also attempting to reclaim available slack time after the task completes according to the procedure in Sec. VIII.

We ran each version over each of our 10 synthetic GRBs, using subsets of the generated gamma rays with sizes 10^N for N from 2 to 6. For each of the resulting 50 subsets, we evaluated the pipeline with a sufficiently large deadline to guarantee an uncompressed state, then imposed deadlines of

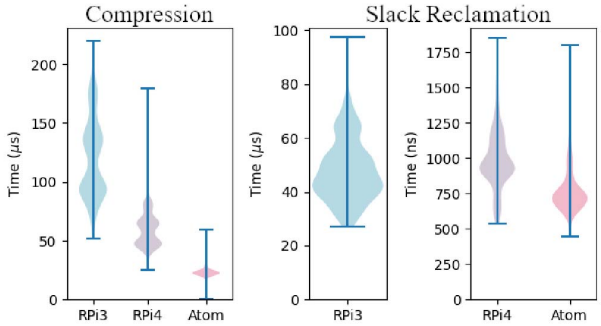


Fig. 9: Measured overhead times.

GRB Catalog #	Duration (s)	α	$E_{peak}(keV)$	β	Fluence (MeV/cm^2)	# Gamma Rays	θ	ϕ
80905499	0.704	0.66	284.6	-2.15	0.918	299 288	33.244	120.90
81209981	0.320	-0.67	1057.0	-2.25	2.452	818 489	40.576	43.60
90227772	0.704	0.48	2013.0	-3.15	20.272	1 772 628	16.766	37.64
90429753	0.832	-0.28	178.3	-1.65	2.643	803 322	31.572	214.17

TABLE III: Simulated short GRBs with parameters matching corresponding catalog entries in [36].

10, 33, 100, 330, and 1000 ms,[2] for a total of 300 sets of inputs to the pipeline. For each set of inputs, we ran **Pareto** and **IntExt** once and ran **Reclaim** 5 times to account for variations in remaining slack time. We observed that over 1750 deadline-constrained runs on each of our three hardware platforms, no instance of the task missed its deadline.

To predict which approach is most likely to perform best on real-world datasets, we compared each approach pairwise with the other two. We define better utility as exceeding a 10% reduction in localization error; we use this threshold to characterize the difference because even a small change in input can result in significantly different results, as reflected by the wide distributions illustrated in Fig. 2. Results for all 1500 runs of **Reclaim** and 300 runs of **IntExt** and **Pareto** are illustrated in Fig. 10. We note that a pairwise comparison provides more detail than enumerating the times each approach is the best of the three, which would not capture situations where two methods dominate the third, but not each other. We also observe that slack reclamation occasionally degrades results, as the additional input events selected might be incorrectly reconstructed or reflect noisy measurements. Nonetheless, the results suggest additional interpolation or extrapolation from an initial candidate state, and reclaiming slack at the end of execution, are expected to improve outcomes most of the time.

C. Evaluation on Short GRBs Observed by Fermi GBM

To characterize how well our approach extends from synthetic training data to real-world workloads, we simulated four additional GRBs sourced from the Fermi GBM catalogs. We used the data in [36], which fits spectral-energy distributions to GRBs observed by the GBM. We searched for short GRBs (duration $<$1s) fit to a Band function; four matched these criteria. Simulation parameters are listed in Table III. We randomly generated source directions by sampling the polar angle uniformly from $0{-}60°$ and the azimuth from $0{-}360°$. Worst-case response times on each platform to perform uncompressed localization of each GRB are listed in Table IV.

Device	80905499	81209981	90227772	90429753
RPi3	1087	1379	5813	1177
RPi4	490	618	2808	529
Atom	265	346	1268	291

TABLE IV: Worst-case response times (ms) for uncompressed localization.

We ran each version of our pipeline for each new GRB. We used a sufficiently large deadline to guarantee an uncompressed state, then imposed a deadline equal to the burst's duration, and finally iterated over the same deadlines evaluated for our synthetic GRBs (10, 33, 100, 330, and 1000 ms). For each deadline, we ran each version of the pipeline 20 times over each GRB. We then found the 68% containment of error in source direction for each set of 20 results. None of the 1440 deadline-constrained runs on each of our three hardware platforms missed its deadline.

Similarly to the analysis for the synthetic GRBs, we compared **Pareto**, **IntExt**, and **Reclaim** pairwise with the other two, enumerating how often each outperformed the others. Results are illustrated in Fig. 11. Counts are out of 28. The results validate our predictions from the initial analysis of synthetic GRBs: interpolation and extrapolation from an initial Pareto-optimal state, then reclaiming slack time at the end of the pipeline, both typically improve localization accuracy.

In Fig. 12, we provide a plot for each simulated GRB of the 68% containment of source direction error in degrees over the 20 iterations of **Reclaim** for each imposed deadline. We observe that, with compression, the APT localization task is often able to produce results close in accuracy to those of its uncompressed state for deadlines of around 100 ms, even

[2]The shortest-duration burst captured by GBM was around 10 ms [29]–[32].

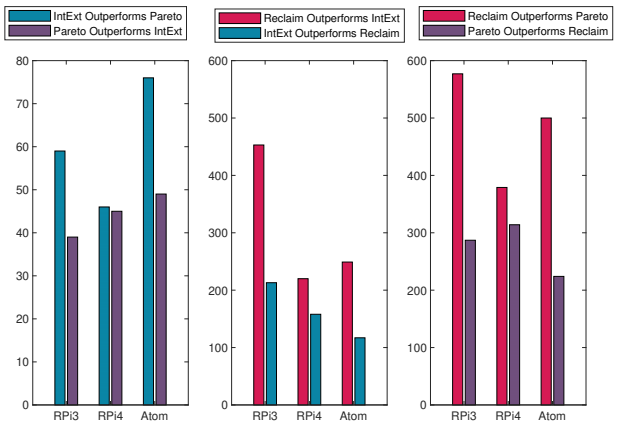


Fig. 10: Pairwise comparison of approach versions for synthetic GRBs.

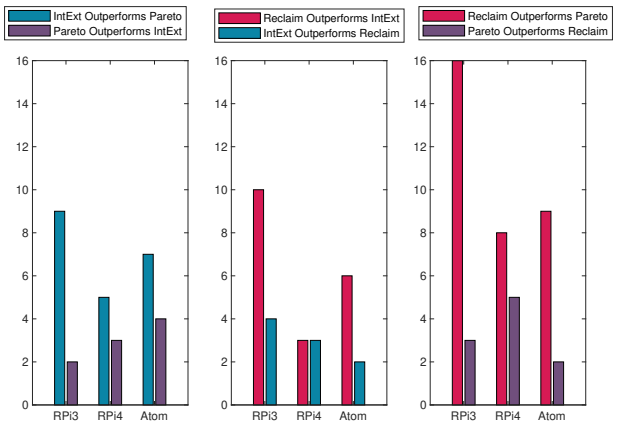


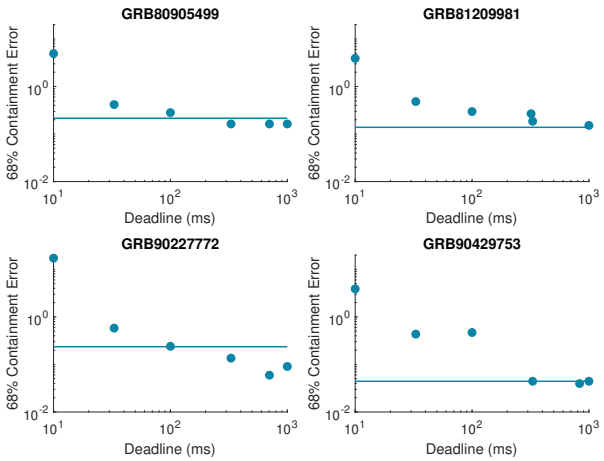
Fig. 11: Pairwise comparison of approach versions for cataloged GRBs.

with worst-case uncompressed response times approaching 6 seconds on the platforms tested. While we allow for further compression to guarantee schedulability in response to dynamic workloads and deadlines, our approach also allows us to characterize a minimum acceptable deadline for each hardware platform. For deadlines as short as 33 ms, it is often successful in providing sub-degree accuracy, sufficient for follow-up observations by optical telescopes. However, as the imposed deadline deadline increases, localization error typically decreases, yielding greater utility. Nonetheless, this is not always the case: because of the high variance in localization accuracy, and because larger values of n_r might cause noisy or incorrectly reconstructed events to be selected, occasionally larger deadlines correspond to lower accuracy.

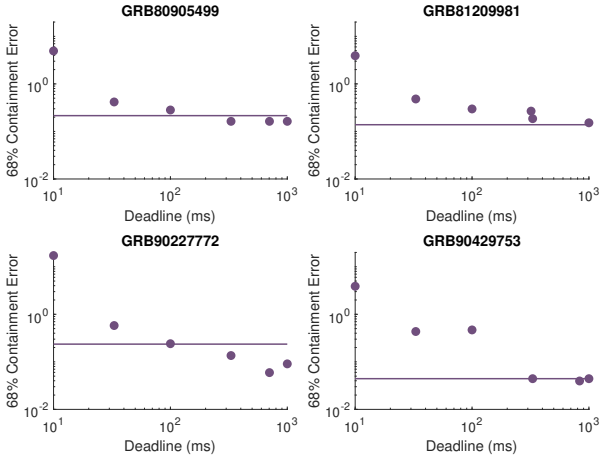
X. Conclusions and Future Work

In this paper, we proposed a technique for compressing the workloads of highly parallel fork-join tasks executing on a fixed number of processors to remain schedulable in the face of dynamic workloads and deadlines. By identifying multiple discrete numeric, continuous numeric, and categorical parameters over which subtask workloads can be compressed, and through offline characterization of their effects on result utility and response times, a Pareto-optimal surface can be generated to enable efficient online compression that guarantees schedulability while minimizing the resulting loss. We also identified methods to reduce pessimism by reclaiming available slack if execution completes early. We demonstrated that, when applied in the context of real-time GRB localization aboard the planned APT satellite mission, our approach provided sub-degree estimates of source direction even for ≈33 ms deadlines imposed by bright, transient bursts.

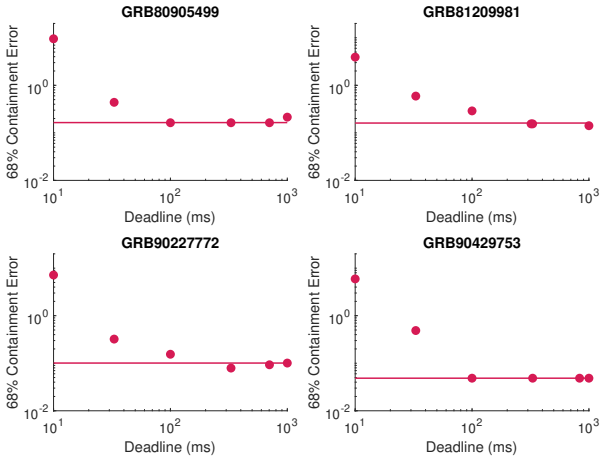
Nonetheless, APT and other space-based missions may present additional challenges that motivate further exploration. The real-time properties of GRB localization might be better expressed with time utility functions, rather than a hard deadline: narrow observation windows may impose a tradeoff favoring earlier, but potentially less accurate, alerts [7]. Additionally, the GRB localization pipeline may run on shared hardware with mission-critical instrument control tasks (e.g., that regulate power or cool the instrument). Alternative analytical frameworks, such as semi-federated [43] or reservation-based federated [44] might allow these techniques to be extended to general parallel DAG tasks that share cores with low-utilization workloads. Additionally, further work is needed to efficiently guarantee a Pareto-optimal solution even when the Pareto-optimal surface does not intersect the region described by the dynamic constraints for a given job. This work serves as a prerequisite toward a utility-driven elastic scheduling model over multiple tasks that share a limited set of resources.



(a) RPi3



(b) RPi4



(c) Atom

Fig. 12: 68% containment of error in source direction using **Reclaim**. Horizontal lines indicate 68% containment for uncompressed execution.

REFERENCES

[1] W. Kywe, D. Fujiwara, and K. Murakami, "Scheduling of image processing using anytime algorithm for real-time system," in *Proc. of 18th Int'l Conf. on Pattern Recognition*, vol. 3, 2006, pp. 1095–1098.

[2] J. Liu, W.-K. Shih, K.-J. Lin, R. Bettati, and J.-Y. Chung, "Imprecise computations," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 83–94, 1994.

[3] A. Soyyigit, S. Yao, and H. Yun, "Anytime-Lidar: Deadline-aware 3D object detection," in *Proc. of IEEE 28th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2022, pp. 31–40.

[4] J. Leonard and H. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *Proc. of IEEE/RSJ Int'l Wkshp. on Intelligent Robots and Systems*, vol. 3, 1991, pp. 1442–1447.

[5] A. Li, H. Liu, J. Wang, and N. Zhang, "From timing variations to performance degradation: Understanding and mitigating the impact of software execution timing in SLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[6] J. Buckley *et al.*, "The Advanced Particle-astrophysics Telescope (APT) Project Status," in *Proc. of 37th International Cosmic Ray Conference — PoS(ICRC2021)*, vol. 395, Jul. 2021, pp. 655:1–655:9.

[7] M. Sudvarg, J. Buhler, R. Chamberlain, C. Gill, and J. Buckley, "Work in Progress: Real-Time GRB Localization for the Advanced Particle-astrophysics Telescope," in *Proc. of 15th Wkshp. on Operating Systems Platforms for Embedded Real-Time Applications*, Jul. 2022, pp. 57–61.

[8] G. C. Buttazzo, G. Lipari, and L. Abeni, "Elastic task model for adaptive rate control," in *Proc. of IEEE Real-Time Systems Symposium*, 1998.

[9] G. C. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni, "Elastic scheduling for flexible workload management," *IEEE Transactions on Computers*, vol. 51, no. 3, pp. 289–302, Mar. 2002.

[10] T. Chantem, X. S. Hu, and M. D. Lemmon, "Generalized elastic scheduling," in *Proc. of IEEE International Real-Time Systems Symposium*, 2006, pp. 236–245.

[11] ——, "Generalized elastic scheduling for real-time tasks," *IEEE Transactions on Computers*, vol. 58, no. 4, pp. 480–495, Apr. 2009.

[12] J. Orr and S. Baruah, "Multiprocessor scheduling of elastic tasks," in *Proc. of 27th International Conference on Real-Time Networks and Systems*. ACM, 2019, pp. 133–142.

[13] J. Orr, C. Gill, K. Agrawal, J. Li, and S. Baruah, "Elastic scheduling for parallel real-time systems," *Leibniz Transactions on Embedded Systems*, vol. 6, no. 1, p. 05:1–05:14, May 2019.

[14] J. Orr *et al.*, "Elasticity of workloads and periods of parallel real-time tasks," in *Proc. of 26th International Conference on Real-Time Networks and Systems*. ACM, 2018, pp. 61–71.

[15] J. Orr, J. C. Uribe, C. Gill, S. Baruah *et al.*, "Elastic scheduling of parallel real-time tasks with discrete utilizations," in *Proc. of 28th International Conference on Real-Time Networks and Systems*. ACM, 2020, pp. 117–127.

[16] J. Li, J. J. Chen, K. Agrawal, C. Lu, C. Gill, and A. Saifullah, "Analysis of federated and global scheduling for parallel real-time tasks," in *Proc. of 26th Euromicro Conference on Real-Time Systems*, 2014, pp. 85–96.

[17] Y. Bai, L. Li, Z. Wang, X. Wang, and J. Wang, "Performance optimization of autonomous driving control under end-to-end deadlines," *Real-Time Systems*, vol. 58, no. 4, pp. 509–547, Dec 2022.

[18] J. Real and A. Crespo, "Mode change protocols for real-time systems: A survey and a new proposal," *Real-Time Systems*, vol. 26, no. 2, pp. 161–197, 3 2004.

[19] A. Block, B. Brandenburg, J. H. Anderson, and S. Quint, "An adaptive framework for multiprocessor real-time systems," in *Proc. of Euromicro Conference on Real-Time Systems*, 2008, pp. 23–33.

[20] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 12 689–12 697.

[21] I. Gog, S. Kalra, P. Schafhalter, J. E. Gonzalez, and I. Stoica, "D3: A dynamic deadline-driven approach for building autonomous vehicles," in *Proc. of 17th European Conf. on Computer Systems*, 2022, p. 453–471.

[22] M. Stigge, P. Ekberg, and W. Yi, "The fork-join real-time task model," *SIGBED Rev.*, vol. 10, no. 2, p. 20, Jul. 2013.

[23] Q. Wang and G. Parmer, "FJOS: Practical, predictable, and efficient system support for fork/join parallelism," in *Proc. of 19th Real-Time and Embedded Technology and Applications Symp.*, 2014, pp. 25–36.

[24] M. Sudvarg, J. Buhler, J. H. Buckley, W. Chen *et al.*, "A Fast GRB Source Localization Pipeline for the Advanced Particle-astrophysics Telescope," in *Proc. of 37th International Cosmic Ray Conference — PoS(ICRC2021)*, vol. 395, Jul. 2021, pp. 588:1–588:9.

[25] J. Wheelock, W. Kanu, M. Sudvarg *et al.*, "Supporting multi-messenger astrophysics with fast gamma-ray burst localization," in *Proc. of IEEE/ACM HPC for Urgent Decision Making Workshop*, Nov. 2021.

[26] I. Bartos and M. Kowalski, *Multimessenger Astronomy*, ser. 2399-2891. IOP Publishing, 2017.

[27] A. Neronov, "Introduction to multi-messenger astronomy," in *Journal of Physics: Conference Series*, vol. 1263, no. 1. IOP Publishing, 2019.

[28] P. Mészáros, D. B. Fox, C. Hanna, and K. Murase, "Multi-messenger astrophysics," *Nature Reviews Physics*, vol. 1, no. 10, pp. 585–599, 2019.

[29] D. Gruber, A. Goldstein, V. W. von Ahlefeld *et al.*, "The Fermi GBM gamma-ray burst spectral catalog: Four years of data," *The Astrophysical Journal Supplement Series*, vol. 211, no. 1, p. 12, Feb. 2014.

[30] A. von Kienlin, C. A. Meegan, W. S. Paciesas *et al.*, "The second Fermi GBM gamma-ray burst catalog: The first four years," *The Astrophysical Journal Supplement Series*, vol. 211, no. 1, p. 13, Feb. 2014.

[31] P. N. Bhat, C. A. Meegan, A. von Kienlin *et al.*, "The third Fermi GBM gamma-ray burst catalog: The first six years," *The Astrophysical Journal Supplement Series*, vol. 223, no. 2, p. 28, Apr. 2016.

[32] A. von Kienlin, C. A. Meegan, W. S. Paciesas *et al.*, "The fourth Fermi-GBM gamma-ray burst catalog: A decade of data," *The Astrophysical Journal*, vol. 893, no. 1, p. 46, Apr. 2020.

[33] P. W. A. Roming, T. E. Kennedy, K. O. Mason *et al.*, "The Swift ultra-violet/optical telescope," *Space Science Reviews*, vol. 120, no. 3, pp. 95–142, Oct. 2005.

[34] W. Chen, J. Buckley, S. Alnussirat *et al.*, "The Advanced Particle-astrophysics Telescope: Simulation of the Instrument Performance for Gamma-Ray Detection," in *Proc. of 37th Int'l Cosmic Ray Conference — PoS(ICRC2021)*, vol. 395, 2021, pp. 590:1–590:9.

[35] S. Agostinelli, J. Allison, K. Amako *et al.*, "Geant4 — a simulation toolkit," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 506, no. 3, pp. 250–303, 2003.

[36] L. Nava, G. Ghirlanda, G. Ghisellini, and A. Celotti, "Spectral properties of 438 GRBs detected by Fermi GBM," *Astronomy & Astrophysics*, vol. 530, p. A21, Apr. 2011.

[37] D. Band *et al.*, "BATSE observations of gamma-ray burst spectra. I. spectral diversity," *Astrophys. J.*, vol. 413, p. 281, Aug. 1993.

[38] "Overview of the Fermi GBM," https://fermi.gsfc.nasa.gov/ssc/data/analysis/documentation/Cicerone/Cicerone_Introduction/GBM_overview.html, National Aeronautics and Space Administration Goddard Space Flight Center, Jan. 2020, curated by J.D. Meyers. Accessed: 26 Oct, 2022.

[39] S. Boggs and P. Jean, "Event reconstruction in high resolution Compton telescopes," *Astronomy and Astrophys. Supp. Series*, vol. 145, no. 2, pp. 311–321, 2000.

[40] V. Connaughton *et al.*, "Localization of gamma-ray bursts using the Fermi gamma-ray burst monitor," *The Astrophysical Journal Supplement Series*, vol. 216, no. 2, p. 32, Feb. 2015.

[41] T. Blass, A. Hamann, R. Lange, D. Ziegenbein, and B. B. Brandenburg, "Automatic latency management for ROS 2: Benefits, challenges, and open problems," in *Proc. of IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2021, pp. 264–277.

[42] M. Sudvarg and C. Gill, "A concurrency framework for priority-aware intercomponent requests in CAmkES on seL4," in *Proc. of IEEE 28th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2022, pp. 1–10.

[43] X. Jiang, N. Guan, X. Long, and W. Yi, "Semi-federated scheduling of parallel real-time tasks on multiprocessors," in *Proc. of IEEE Real-Time Systems Symposium (RTSS)*, 2017, pp. 80–91.

[44] N. Ueter *et al.*, "Reservation-based federated scheduling for parallel real-time tasks," in *Proc. of IEEE Real-Time Systems Symposium (RTSS)*, 2018, pp. 482–494.