Dynamic Orchestration of Data Pipelines via Agentic AI: Adaptive Resource Allocation and Workflow Optimization in Cloud-Native Analytics Platforms

[1]Hara Krishna Reddy Koppolu, [2]Anil Lokesh Gadi, [3]Shabrinath Motamary, [4]Abhishek Dodda, [5]Sambasiva Rao Suura

[1]Data Engineering Lead, CSG Systems International, harakrishnareddyk@gmail.com
[2]Cognizant Technology Solutions - US Corp, anillokeshg@gmail.com
[3]Software Systems Architect, Saturn Business systems inc, motamaryshabrinath@gmail.com
[4]Sr AI Solutions Engineer, Centific.com, abhishek.dodda9@gmail.com
[5]Sr Integration Developer, Natera Inc, suurasambasivarao@gmail.com

Abstract: The move towards more dynamic machine learning (ML) is present in all sorts of areas within the field, creating a tantalizing possibility to use such methods to not only create models that are more in tune with the constantly changing world we live in and the data which is being generated from it but also a world where the automated pipeline processes behind the ML modeling are severely underutilized. One area that has received little attention in this burgeoning area is most of the work being done under the umbrella of agentic AI, or methods aimed at creating agents that are semi-autonomously able to do complex tasks with little human interaction. This paper aims to summarize some of the work, both in terms of completed tasks as well as the technologies that will allow us to create a new paradigm for an automated data ML pipeline, both in terms of how data is consumed for ML tasks as well as how tasks are set up, monitored, and completed. Using an agent-centric approach we seek to allow agents to both have different expertise levels with regards to different types of ML tasks, be able to learn over time by themselves which are the most competent agents for certain tasks, as well as having users not need to worry about piping together a complex set of steps for a complicated task. Rather the user will simply provide a high-level description of what they need accomplished, as well as any constraints, and the agent system will find the best pathway for accomplishing the task and either execute the pathway itself or coordinate with other agents to complete the task. Such a paradigm will allow for the automated orchestration of very complicated tasks as well as the execution and monitoring of larger, often cumbersome and fragile task pathways, and allow domain specialists for any field to be able to utilize ML pipelines to accomplish their goals while the technical details are handled by specialized ML agents.

Keywords: Agentic Machine Learning, Automated ML Pipelines, Dynamic Learning Systems, Task-Oriented AI Agents, Semi-Autonomous Agents, Agent Expertise Modeling, Pipeline Orchestration, Constraint-Based Task Execution, ML Workflow Automation, Multi-Agent Coordination, Task Monitoring and Adaptation, High-Level User Instructions, Adaptive Agent Learning, Intelligent Workflow Optimization, Data Consumption Automation, ML Task Abstraction, Scalable AI Infrastructure, Fragile Pathway Management, Domain-Agnostic AI Interfaces, Autonomous Data Science Systems.

1. Introduction

Much of the research and practice to date has focused on the design and execution of individual, one-off analytic functions. However, defining and executing an entire data pipeline in the style of a DAG is mostly static, requiring a manual orchestration of many individual analytics that transfer intermediate analytics data among each other, manage mutual dependencies, and create a comprehensive solution for a complex business analytic question at hand. Even the execution of data pipeline DAGs is often not driven by a combination of pipeline DAG dependency relations and pipelined-data readiness to be consumed. Instead, pipelines are often executed by their dependent analytic function data transfer specified by a schedule or specific conditional triggers. Further, the focus is often on just optimizing performance, reliability, and error handling of the execution of individual pipelines.

It is also increasingly the case that organizations want to operationalize the building and execution of complex analytics data pipelines to generate business value on an ongoing basis. They want the entire data pipeline process to be automated and augmented. The research for cyber-physical systems has defined high-level objectives for the envisioned solutions. They should be dynamic, distributed, modular, multi-function orchestration approaches using lightweight embedded control threads. Such agentic compute control is also a key characteristic of autonomous intelligent agents, emphasizing autonomy, intelligence, and reactivity.

In this report, we present our research towards this vision of data pipeline agentic control. In particular, we describe the functionalities and architecture of a cloud-based platform we are building to support agentic data pipeline management. The dashboard of this platform allows data scientists to focus on defining what the pipeline should do and when, while the agentic AI pipeline orchestrator supports their ongoing automated, intelligent execution. The approach also allows for the transparent incorporation of human insight into the AI decision processes during execution. The framework outlined here is provided for broader use.

1.1. Summary of Key Points and Objectives

There are ample domains where well-established data dependencies and synthesis methods allow for the straightforward crafting of complex workflows, provided we have the right tools for low-burden specification. A parallel can be drawn with declarative user interfaces. In simple user interfaces, users are guided throughout their interaction by the application, which takes on responsibility for low-level concerns, while the user specifies the high-level problem they want to solve. With declarative interfaces, users fill in the missing pieces and the interaction is left open-ended. This simple act can relieve users of extensive work and cognitive burden. For example, in spreadsheet software, users can specify complex computation distributions with simple formulas embedded in each cell, as opposed to controls and buttons for each operation of the workflow.

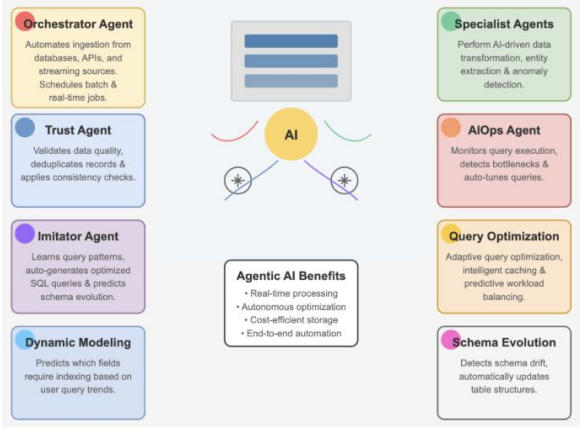

Fig 1: Agentic AI for Data Management

This document examines an approach to declarative interfaces for complex computational workflows, considering not the workflow itself but the underlying data dependencies and standard components. We focus specifically on data dependencies and pipeline modularities that have previously been used to implement and execute complex data pipelines locally. Intermediates and tools for resolving far-reaching dependencies at each stage of a pipeline can greatly relieve users from the substantial burden of piecing together collections of micro-scripts, especially when their applicability is different for every

input, as is common when using functions designed to rely on low-level programmatic structures. We consider not only simplifying the user burden of such pipelines, in terms of setup and external calls made during execution but also addressing one of the main shortcomings of these automated data pipelines to date; specifically that very few have used an agentic form of AI, guided by claimed motivations, to provide a user-facing interface.

2. Background and Motivation

Data is often described as the new oil, driving our world forward with its inherent value. While the oil analogy certainly fits the massive, latent value of data, it tends to overlook that data processing can often be harder than data collection. Systematic, scalable data processing enables organizations to extract value from the data that they collect, by literally creating intelligent services and products that can provide enormous value. Modern society is essentially powered by such products and services, and the digital economy rests on a foundation of ad-hoc data collection and frequently broken data processing pipelines. These pipelines are ideally data-driven but typically require extensive effort to be designed, implemented, and monitored by highly-trained technical staff. In recent years, considerable resources have been invested in making it easier for companies to build and deploy machine learning models without in-depth technical expertise. But similar support around automating the design and operations of data pipelines is still nascent.

Ad-hoc, brittle data pipelines are maintained in a state of fragile despair by overworked engineers, from the days of simple hourly cron jobs to the modern era of orchestrated micro-batch and stream processing frameworks. Today's state-of-the-art tools provide extensible orchestration systems to tie together diverse data processing components. Components written in different languages, encode different forms of processing, are executed in different runtime contexts, log different kinds of outputs, are integrated into pipelines with their own data interfaces, and are scheduled and monitored with their workflow management configurations.

$$\hat{R}_t = \sum_{i=1}^{n} \alpha_i \cdot x_i(t) + \epsilon$$

where:

- $\hat{R}_t$ = Predicted resource demand at time t
- $x_i(t)$ = Feature input (e.g., task size, data volume, latency)
- α_i = Learned weight for each feature
- ϵ = Noise/residual error term

Equation 1: Resource Demand Forecasting (RDF):

2.1. Rationale and Inspiration Behind the Study

This research is motivated by two observations. First, modern data scientists, especially in industry settings, spend a significant part of their time and effort on what is called data engineering, as opposed to data science. Despite being critical for the success of any data science effort, the engineering aspect of working with data is overly manual, labor-intensive, and dependent on ad-hoc solutions. This represents a new frontier where AI can have a great impact and should be equipped to make it easier to perform tedious tasks and automate what is feasible to be automated.

Second, advances in Agentic AI have reached a point where workflows in multiple-step interactions and usage of multiple tools are becoming possible. In particular, the most recent models demonstrated snowballing capabilities in completing tasks that they are instructed to perform, including careful and accurate use of a large number of external tools, and the ability to call multiple chains of functions as part of interaction with external programs needed to assist users in achieving their goals. With the domain of data engineering being such a good target since it has many tasks that can be performed modularly in a pipeline-like way and many AI and non-AI tools that provide programmatic access to be orchestrated via a suitable wrapper, Agentic AI offers a promising new approach based on dynamic orchestration of data engineering pipelines to help data scientists spend more time on data science, and less on data engineering. So we wonder: How much can Agentic AI help with data engineering?

3. Agentic AI in Data Pipelines

Although traditional AI has made substantial advancements in recent years, we're nevertheless still grappling with significant data challenges. Moreover, the potential of AI in data processing has not been wholly realized. The crux lies in the fact that pipelines of today mostly rely on static orchestration, with little higher-level management beyond the scope of demonstrating pipelines to provide results. These traditional AI capabilities function well in the near term, but numerous enterprise-scale barriers can block innovation. These barriers stem from a combination of factors, from misaligned enterprise processes to the sheer scale and complexity of enterprise systems. Yet, these systems still need to be on-boarded, integrated with broader enterprise ecosystems and resources, trained, and validated. The result is an overwhelming transactional burden at a crucial stage of modern innovation; the lack of sufficient investment in foundational middleware infrastructure needed to facilitate automation of these undertakings has blunted the advantages afforded in innovation timescale, cost, and efficiency. Pipelines are traditionally viewed as ways to prepare data, train models, and serve results. Advanced AI techniques can, and should, go further. Rather, data pipelines should be thought of as dynamic, constantly evolving data processing systems that should also seek to discover new ways for the enterprise to leverage its data. Various intermediate stages of data processing can benefit from AI functionality as well, creating an overall four-tiered structure of data preparation, model training, result provisioning, and data innovation. These stages are sufficient to address the function of needing a data pipeline capability to support the scaling out of AI capabilities, but not necessarily the level of agentic capabilities needed to satisfy the statement of requirements that requested the demonstration of advanced decision-making capabilities that crossed traditional domain boundaries. The remaining two pillars of Agentic AI lie in the continuous discovery of enterprise knowledge and the high-level management of system performance through goal-setting and plan deployment.

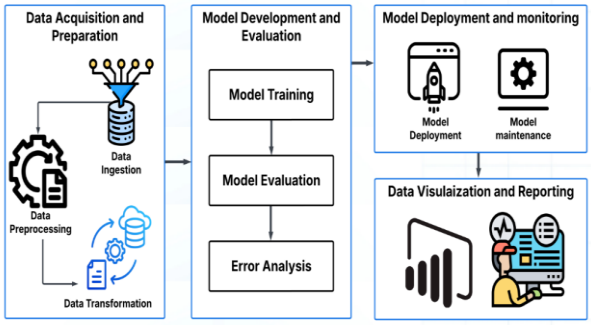

Fig 2: Integrating AI-Powered Agents for Data Pipeline Optimization

3.1. Understanding Agentic AI's Role in Data Processing Pipelines

Environments inhabited by collective decision-making agents, decentralized agentic systems, or dynamical multi-agent systems, are virtually ubiquitous. Collective decision-making by multiple agents is one of the most mature domains in AI and has had astonishing practical results ranging from market economics to molecular machinery. Here we would like to detail how AI has reached a point where it offers agentification as a cohort for the different approaches. In terms of the use of AI as a component in data processing pipelines, the integration of multiple systems into a compelling solution is not an easy feat. AI systems have a lot of capacity to synthesize and code the application space into coherent applications, but they are not, at least not yet, universal resourceful architects.

The problem of assembling decision-making skills majorly diverges from the logic of the agentic solution process because while the individual modules behave as serfs or slaves that act to optimize the expected responses of the users, the different modules - the AIs and the users - have access to different individual Private Experiences in their decision-making task, yet to put together an ensemble of them that offers the flexibility, power, and robustness required for practical data processing solution is a delicate task. Instructions have to be entered or implemented following their particular vocabularies. Refusal of service or preference divergences have to be handled in a way that steps back from practical solutions to dynamic back-and-forth buttering of agentic solutions or optogenetic tweaking is to be handled deftly and almost on the fly to make the whole composition credible.

4. Cloud-Native Analytics Platforms

The advent of cloud-native technologies has triggered a shift in architectural design patterns among enterprise technology vendors. In the realm of big data, these technologies have empowered a new breed of analytics platforms that are built on flexible services and micro-services to leverage the cloud's elasticity. Row and document stores, columnar databases, stream processing, batch processing, user interfaces, and data storytelling can all be delivered through cloud-native architectures using commodity storage, processing, orchestration, and relational services centered around data. Most well-known analytics platforms have adopted hybrid approaches, offering a broader set of technology options. What is of significance to our research is the acceleration in architectural innovation among new microservice-based startups that are growing faster than their established counterparts. Investors have encouraged these new vendors to build solution stacks based on microservice abstractions.

The architecture of a cloud-native analytics platform is tightly aligned with the complex, ad hoc decision trees used by analysts and business users to extract insights from data. As a result, these analytics platforms can automate higher-fidelity analytics insights, paving the way for organizations to realize the vision of data-driven decision-making. Although the business insights generated by data pipelines are often non-iterative, they can be expensive to mix. In AI, large cost advantages are possible at multiple levels in the analytics stack. Agentic AI, with the potential to dynamically optimize data pipelines — both the selection of individual algorithms and the underlying computational services that perform the computations — offers the prospect of making AI even more cost-effective. Agentic AI makes data analytics pipelines more efficient by organizing the data and the computational resources in a preparation step before running the analysis for insight discovery.

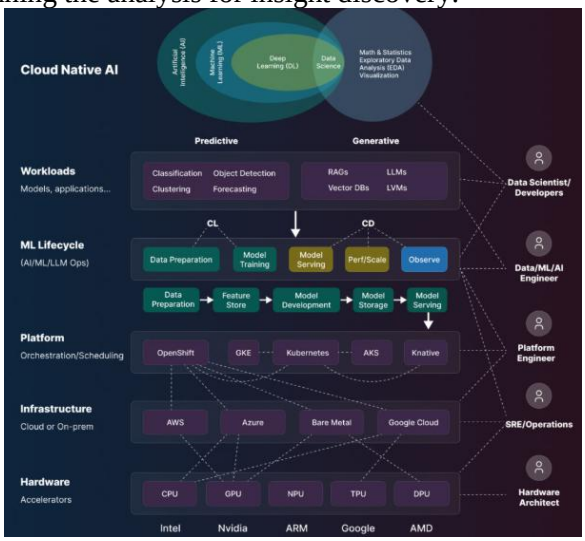

Fig 3: Cloud Native AI

4.1. The Function of Agentic AI in Enhancing Data Pipeline Efficiency

Cloud-Enabled Analytics Platforms are tools of choice to perform analysis over heterogeneous and large-scale data. In addition to streamlining the analysis of raw data ingested in data lakes, they also offer new evolutionary paths for existing data warehouses. However, the design of the workflow for high-level analysis requests is not yet a solved problem as these platforms expose a complex set of APIs and optimizers in a wide variety of backends. Experts typically need to use multiple languages, implement orchestrations, and do reverse ETL between functionality silos. Reducing the effort in defining analysis tasks in a platform-agnostic way is a substantial enabler for making analytics accessible to the broader range of knowledge workers. Enabling the semantic composition of tasks lowers the barriers to entry for collaborative analytics.

Agentic AI performs explicit functions to enhance the efficiency of data pipeline execution: primarily composition, scheduling, and failure recovery functionality. Composition performs multi-level abstract syntax tree transformations and concrete to abstract decompositions. Scheduling using heuristic search over execution models and response sensitivity can enable platform-aware optimization. Proto-C is an expansive agent-based and programmatic approach to address many of the hurdles common for low-code data ingestion and cleaning tasks distinctive during preparation for analysis. Template expansion with implicit input/output semantic models can assist generalization to novel tasks. The participative

assistance provided by Agentic AI generally works best for routine tasks for which a sufficient number of precedents are available. Unlike compiler-based programming paradigms, it is always possible to seed experience-based systems with a small percentage of intelligence, collocated alongside much larger computation or data exchange requirements.

5. Dynamic Orchestration Framework

1. Definition and Scope

We define dynamic orchestration of data science as a family of capabilities encompassing the orchestration of diverse components with online discovery, replanning of pipelines upon change, adaptive execution handling uncertainty, and task coordination among heterogeneous agents or AIs. We restrict discussions to orchestration primarily focused on data pipelines rather than the broader view encompassing applications such as smart factories or robotic interactions with the physical environment; and the shorter-term predictions of needs with mechanisms for resource allocation and inter-task resource sharing used in some smart factories.

We are not aware of existing systematizations or detailed analyses of existing dynamic orchestration capabilities either at a survey level or based on a specific set of use cases as the definition of data pipelines with agentic AI enables. This work is targeted at the outlined information gaps from the perspective of actionable areas for theoretical and applied work on dynamic orchestration capabilities. It describes capabilities from the viewpoint of a data architect tasked with designing a dynamic orchestration capability over a data pipeline.

2. Key Components

Dynamic orchestration capabilities fall into five core capabilities tentatively framed as questions, opening a pipeline designer's toolkit. Understanding these capabilities and their implications for agentic AI was a premise for the use case conceptualization of pipelines with agentic AI. Online discovery is what the agents do when no planning or replanning support is available: browse, reactively respond via monitoring, or query a knowledge base. What role do formal and informal knowledge representations, natural language, transfer learning, world models, and causal models play? Replanning of workflows upon changes and the triggers for this replanning are key. What manual/agentic coordination and quality assurance mechanisms across workstreams and agents are needed?

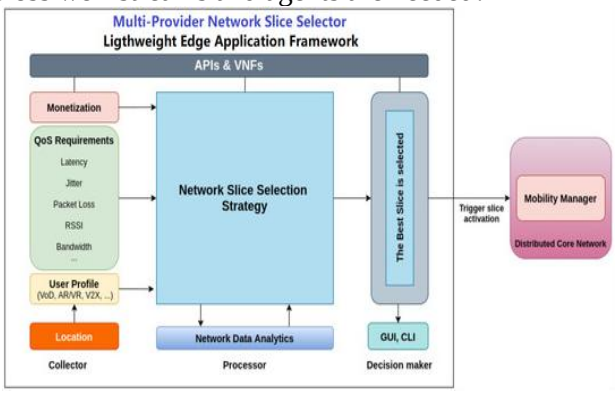

Fig 4: Dynamic Orchestration Architecture

5.1. Definition and Scope

In this section we introduce the concept of Dynamic Orchestration in a form to ground the ensuing discussion. We differentiate between static and dynamic orchestration approaches, speculated on possible use cases of Dynamic Orchestration, and summarize current work.

In the original sense, orchestration is about correctly sequencing a set of activities during their execution. Orchestration defines control dependencies among a set of activities. A central controller with a full view of the overall execution quality exerts control of the set of activities. In this sense, any current approach that manually or conditionally controls a closed set of activities is committing orchestration. Any Intelligent Workflow manager is doing orchestration.

But do traditional workflow executives fit under this definition? Volume and topology changes modulate the control dependencies; local controllers execute each activity separately; no configuration is possible before task creation. They do not exert control over sub-activity decision-making but rather rely on external dataset-triggered rules to initiate actions. These graphical workflow simulations of modular systems handle yaw well with cumulative task-by-weight dependency – the only phenomena that govern

task creation. Under volume and topology changes, optimization of the mixed state of the cellular automaton, although known to be off-execution, only provides us with a rough control on how regular the resistivity will be and how much funneling will result – guide task creation triggers. Thus, workflow executives cannot help here, and technologies like reduction, modular simulation, macro operations, or hybrid approaches, which allow for ad hoc supplementation, are necessary.

$$A_{i,j}(t) = \frac{U_{i,j}(t)}{\sum_k C_{i,k}(t)}$$

$A_{i,j}(t)$ = Allocation score of agent i for resource j

$U_{i,j}(t)$ = Utility of allocating j to i (based on speed, load, priority)

$C_{i,k}(t)$ = Estimated cost of using resource k by agent i

Equation 2: Agentic Allocation Function (AAF):

5.2. Key Components

The dynamic orchestration framework has four main components: the user, an agentic planner, a set of external language tools, and a set of actors whose capabilities are wrapped in one or more language tools. The user is a person with an agency or a group of people with common pragmatic goals. The agentic planner is an AI agent capable of reasoning, planning, reviewing, and acting upon feedback while executing complex plans. The external language tools are offline and live connections to existing technology that the planner can use to increase its level of intelligence. The set of actors are agents with capabilities that may or may not be agentic. These agents may represent other people or nonhumans with agentic characteristics as interactive language learning machines.

To illustrate these components and possible interactions, we use two graphics. The first shows the basic setup for user-initiated short to medium-term plan agreements. The solution space has four dimensions: the number and characteristics of planners, the number of people and their characteristics for which assistance is being planned, the number of language tools used by the planners to augment intelligence and the relationships among the planners, the people being assisted, the external tools and the other language model agents.

The second graphic provides a more complex view of the diverse interactions involved with the longer view of multi-level hierarchical conversations that connect the user and planner with the tools and the people. Users may provide one or more annotations of a shared task slate, and planners may supplement these task definitions with their agenda directives for the language tools or actors. The tasks being assisted may be small, discreet time-scaled actions that achieve the user goals in the near term, or they may be subgoals intended to provide long or short-term trade-offs as a behind-the-scenes strategy.

6. Resource Allocation Strategies

A potential aspect of these processes is the resource allocation strategy that an orchestrator may adopt to manage the utilization and costs associated with computing resources to optimize the performance and impact of the pipelines. Resource allocation strategies may be broadly classified as static or dynamic based on the frequency with which the allocation decisions may change. Static resource allocation assigns fixed resources to each pipeline execution step and uses these resources for the entire time that the pipeline step is being executed. This allocation may be done based on estimates or learned models that provide predictive information about the cost of executing a pipeline step with various resources. Examples of static allocation would be to micro-manage the pipeline steps to run with the same dedicated resources for their entire execution duration or to schedule the pipeline in such a way that at any given point in time, the pipeline steps in execution would use a combination of available hardware accelerators that has the least estimated cost.

On the other hand, dynamic resource allocation strategies change the assigned resources for a step during its execution to better utilize available resources for better performance or cost efficiency. Dynamic allocation may happen at a step level by frequently adjusting the step-specific allocated hardware resource values based on the changes in execution context observed during execution, or across different pipeline execution steps, in which case the orchestrator is in effect scheduling the pipeline steps for execution just like a resource scheduler does for concurrent step executions. The orchestrator may decide not to exclusively control the allocation decisions during execution, but rather in an opportunistic way when such changes would have the most impact on any individual step execution. Dynamic allocation

may be adopted for pipeline types for which execution duration estimation models do not provide accurate duration estimates.

6.1. Static vs. Dynamic Allocation

Resource management is a fundamental and complex problem of cloud computing. Essentially, allocation strategies can be separated into two very different categories: static allocation and dynamic allocation. Once we make one of these choices in a system, it can be very costly to switch to another.

Static allocation is based on a two-step pre-planning process. The user selects one or multiple service templates describing the constant expected performance of a service, and then, a central orchestrator manages which physical resources need to be pre-allocated to guarantee that the quality of service can be fulfilled. These choices can be made a priori and do not change, independently of whether the demand varies a lot or not. Evaluation of the service cost for users can be based on the templates but the penalty when overstating or understating requirements can be much higher than the monetary cost of the allocated resources. Static allocation is based on a two-step process. First, a bandwidth requirement forecast is performed, over a long period. Second, resources are allocated, based on the forecast.

While attractive in its simplicity, such an approach has major shortcomings. The two steps are separated. First, the request forecast is not precise enough to make optimal allocation decisions, and second, the allocation decisions must be located far before the assignment of services to resources takes place. As a result, over-provisioning is needed. “Locking” physical resources for an entire resource allocation time frame, even supplying some time to the orchestrator for that allocation, is a very wasteful process. In particular, if the resources are very high throughput, such as in video transcoding applications, it may represent a significant share of service time.

6.2. Cost-Effective Resource Management

The allocation of resources to workloads is often a trade-off between the time an individual is willing to wait for completion versus the time required for completion, as well as the cost of execution within a specific time frame. If a user is willing to wait hours longer, many data pipelines may be charged at a lesser price. Cloud providers have complicated price structures, composed of factors such as hourly cost, cost of memory, and availability of discounts. There can be minimum charges, so it may be worth scheduling a few more jobs to fit the minimum time. Resources can be allocated individually according to pipeline time needed and priority, or collectively to phases; that is, if one pipeline in a phase is expected to take 10 minutes, the phase may allocate the resources needed to finish all jobs within a 10-minute window, even if the individual pipelines may take different amounts of time to execute. Due to the existence of transaction times and phase dependencies, that time may be very different for each pipeline, especially with different data sizes. Building an optimal scheduling system is a complicated job; however, the pipeline phase system greatly simplifies this problem.

Developers allocate cloud resources for AI processes that are big consumers of compute time, network, storage, and cloud cost/usage time. For example digital initiatives, this effort requires a lengthy review process to approve and fine-tune a budget large enough to sustain many weeks, months, and even years of cloud storage and computing activity. The cloud cost may be appreciably reduced if additional budget plan time and monitoring with a simple statistical model can predict monthly resource needs. A continuous portion of the effort spent tuning model hyperparameters to reduce prediction error can be cut down with a model whose tuning occurs during periodic idle cycles of the cost prediction model.

7. Workflow Optimization Techniques

Workflows consisting of a sequence of tasks represent a large space of data-centric operations, which benefit from careful orchestration. All popular cloud service frameworks allow users to harness resources with different levels of automation, ranging from manual specification of particular resource configurations and scheduling times to let the framework make those decisions on their behalf. Choosing the right target containers or virtual machines can affect performance and cost directly. Resource elasticity is an important component of costs. When idle, a VM incurs cost without yielding any work. Dragging small tasks takes up space and incurs high latency when task granularity is small. Throughout the lifecycle of a data-centric task, there are opportunities for optimization through scheduling. The choice of a task runner, a configuration frame and a deployment strategy in orchestration platforms has a deep impact on the production systems. With the development of cloud computing, parallel pipelines with multiple running subprocesses can better fulfill time-specific requirements for the delivery of results.

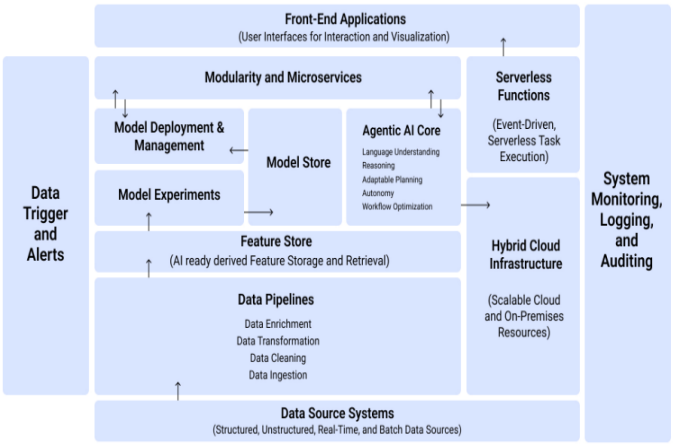

Fig 5: Agentic Workflows

Orchestration platforms also enable optimization of pipelines intended to execute for extended periods. The optimization methods range from simple, heuristic error correction to reinforcement learning and gradient tree boosting problems. This short section looks only at techniques to optimize parallel pipelines. Seasonal climatological differences can impact workloads processed in pipelines. Cloud service providers typically charge for storage and network transfer, in addition to computing and on-demand storage of data results. In this sense, optimizing on-demand interactions with pre-exercised data products annotates expenses incurred during the duration of the interaction. Computing potential savings is a function of estimating the results during the pipeline evaluation. Programmers may be required to run operations in parallel, and the software would produce multiple outputs.

7.1. Scheduling Algorithms

Scheduling goes back to the early days of computing and has traditionally involved the assignment of resources to tasks over time, with the goal of some optimization criterion. In the context of workflows, scheduling algorithms map task and resource characteristics, as well as interdependencies between tasks, which in the case of inter-dependent workflows are error management, data transfer, and delayed execution, onto a list of tasks. These algorithms may use makespan, resource usage, replication overhead, fault tolerance, or cost as their metrics. Heuristic-based approaches are popular in workflow scheduling, where the task list represents the order of task execution based on some optimization criteria. From our admittedly partial view, the main thrust of scheduling in dynamically executing workflows is an attempt to optimize what we characterize as the optimistic case of overall resource load and use in the simplest possible way. It is viewed as a scheduling problem of typically time-extended tasks subject to a user-specified deadline. Just-in-time workflow executions, with the flexibility for resource overloading during critical execution periods, rely heavily on good scheduling; but also enable resource overloading at temporo-spatial clusters. The scheduling algorithms described here support this dynamic reconfiguration of workflows by managing the load on different resources assigned to the jobs in the cluster. Distributed execution of workflows introduces a natural distribution of data dependencies across the execution phases of different workflow instances.

7.2. Load Balancing Approaches

Beyond using scheduling to assign tasks for execution, individual task execution times also have an important effect on minimizing workflow completion time. In particular, when certain tasks take longer to execute, they may hold up the completion of workflow execution. One way of mitigating this problem is to balance the load across parallel resources, thereby reducing the maximum task output time. Several different load-balancing strategies can be employed. Load balancing can be performed statically, where the load is determined a priori based on estimates of the execution duration of tasks on different resources, or dynamically, based on current system status regarding execution time or resource utilization. Additionally, load balancing can be performed opportunistically or consistently. Opportunistic load balancing takes place at execution time, and modifies the resource allocation in an ad-hoc manner, whenever the execution of a task can be switched from one resource to another.

In contrast, a technique is deemed a consistent load-balancing approach if it performs load-balancing systematically. A typical example of a method of load balancing is the division of large input data sets into smaller chunks to have different processes execute on different resources a subset of the work, then merge the output of these different processes when completed. Static load balancing was originally

proposed for use within data-intensive workflows with static plan graphs. Since the execution durations for the different tasks are known a priori, the task execution plan can be obtained by solving a mathematical optimization problem. Static load balancing approaches only produce good results if the estimated execution times are accurate and have a low variance. Dynamic load balancing works better in environments with low accuracy and high variance in execution times, and should thus be used for workflows in more dynamic environments.

8. Performance Metrics

Performance is of high importance for any automation stack and is also a key research direction for the specific case of Data Science tools. In this section, we present the performance metrics that we use to evaluate the dynamic orchestration framework. A crucial aspect regarding the performance metrics is that in the context of the dynamic orchestration of pipelines, two key variables affect performance: (1) the agent's internal configuration and (2) the external workload. The agent's internal configuration summarizes the details of policies that are active, functions that are loaded, and resources that are allocated. The external workload is characterized by the number of requests it handles, their arrival time, and their complexity. Experimental results can be analyzed according to different scenarios. The main metric is performance gain, which estimates the improvement in performance from using a dynamic framework compared to static or hybrid alternatives. This metric can be viewed as other performance measures such as throughput or latency.

Two standard context metrics are latency and throughput. The classical question that defines latency is: how long does it take for my request to be processed after I push the button? In the context of dynamic orchestration of pipelines, latency corresponds to the time that the dynamic framework needs to minimize latency: starting from when a request is first activated until the completion of the request's pipeline. Users usually care about latency, as they are not concerned about how many requests are processed per time unit, but how often they can get final results for their requests. An alternative metric is throughput, which considers only complete tasks without concern for latency. The classical question of throughput is: how many requests are processed in a time unit? Which is usually dismissed by users. However, it can be of interest for Batch AP modeling. The fact that one metric from the other is related to the current size of all requests, as well as to the order in which requests are completed.

8.1. Throughput and Latency

In a data pipeline, throughput is the number of data samples processed in a fixed interval of time. For example, the throughput for image processing pipelines that generate segmentation masks is the number of (input image, output mask) pairs generated in a second. Latency is the duration of time taken to publish an output after receiving the corresponding input. In the above example, the latency is the time duration to generate a mask from an input image. For batch processing pipelines, such as the case when data samples are received in groups, batch time replaces throughput, and all of the data samples in a batch share the same batch latency. Throughput and latency are conflicting metrics to optimize because increasing the latency usually lowers the throughput, and vice versa, hence a Pareto front characterizes the domain of optimal combinations. Low and high latencies have very different effects on downstream tasks, making the distinction important. In a robotic assistant navigating a home, high latency often leads the robot to wait at a room entrance, causing frustration. Low latency is preferred in this case. In contrast, completing object manipulation as fast as possible, regardless of the latency of individual manipulation, is often more beneficial to the user. In such cases, optimizing throughput rather than individual sample latency is more desirable.

Latencies are generally more important than throughputs as performance indicators of behavioral pipelines in AI systems for four reasons. First, since the data processing rates of AI systems are relatively high compared to AI systems in earlier times, throughput is limited mostly by mechanical parts, such as robotic arms or legs moving in physical environments. Second, long latency between any two steps in an AI system introduces perceivable wait periods and frustrating user experience. Third, increased latencies of many components add up, possibly exceeding the durations of relevant user activities. Consequently, the user may stop what they are doing and decide not to continue. Fourth, so far, throughput has not been the dominating metric in the AI community.

8.2. Scalability and Fault Tolerance

Given the multi-agent operation of AIDA, scalability is a paramount performance concern. Like some other autonomous agents that act on data, multiple AIDA agents can act on the same data at overlapping

times. However, unlike validation agents, AIDA agents interact with data pipelines by executing a series of complex transformations having a clear final goal. In addition, pipelines can involve numerous data abstraction types; diverse agent skills, interests, and capabilities; and variably sized data abstractions in massively parallel arrangements. Indeed, a single AIDA agent instance may simultaneously be responsible for agilely coordinating multiple heterogeneous AIDA sub-agent instances at distributed locations, from different service providers. Thus, in terms of anticipated pipeline transformation types, conditions, resources, and timing, our nano-services orchestrate what may exist as countless unique and highly dynamic microservices.

In addition to scalability, AIDA infrastructure must ensure fault tolerance, preventing data loss in most conceivable data supplier events. Such events may arise from data generation and transmission device failures, unexpected behavioral changes in devices, or even faulty data logic. Our proposed policing service infrastructure provides for real-time monitoring of enrolled data pipelines for such anomalous events. In particular, nano services actively synchronize results from redundant AIDA agent instances that continually relay data between origination and reception nodes using the same interface enrolled with the same service provider.

Equation 3: Workflow Optimization Objective (WOO):

$$\min_{\pi \in \Pi} \sum_{t=1}^{T} (\lambda_1 \cdot L_t(\pi) + \lambda_2 \cdot E_t(\pi) + \lambda_3 \cdot \Delta_t(\pi))$$

where:

- $\pi \in \Pi$ = Workflow orchestration policy
- $L_t(\pi)$ = Latency cost at time t
- $E_t(\pi)$ = Energy/resource consumption
- $\Delta_t(\pi)$ = Pipeline disruption or error likelihood
- $\lambda_1, \lambda_2, \lambda_3$ = Tunable trade-off weights

9. Conclusion

In this work, we examined how agentic AI – an advanced and rapidly developing form of AI based on deep learning and reinforcement learning techniques – should take over a significant part of the routine work associated with the development and deployment of data pipelines that are at the center of data-driven operational decision making. In doing that, we explored the central components which together comprise a dynamically orchestrated ecosystem for agentic AI – meta-AI capabilities that enable any domain expert to easily express their objectives, provide solutions in real-time, dynamically manage the systems' ongoing development and operation, and well as facilitate communications and the flow of information on which data pipelines depend, and the agentic AI agents themselves, which perform in parallel much of the work, in an actor-critic architecture based on their own and humans' prior experiences. We concluded that neither an open agent-based architecture for data pipeline orchestrators, nor a task decomposition API for large language model-based agentic AI solution generators, currently exist, but should be implemented as key enablers: Firstly, an API to orchestrate deep agent hierarchies, where each agent's subtask decomposition, resource allocation, and microtask parallelization is automated via a large language model-guided interactive learning process based on a combination of real and simulated experiences, using reward shaping and possibly imitation learning. Secondly, a standard set of capabilities, interfaces, and protocols to support an open actor-critic architecture for these agent-based orchestrators, where the bulk of the solution is generated in parallel by a continuously improving ensemble of parallel worker agents. We believe that these implementations could radically speed up the development of agentic AI for data pipelines, which is to the best of our knowledge not being currently pursued.

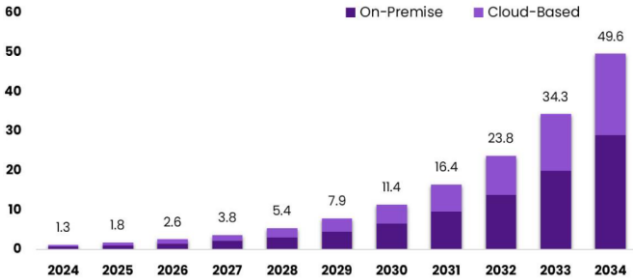

Fig 6: Agentic AI Architecture Market Size

9.1. Future Directions and Implications for Research

The research we describe in this work presents a foundational block for reasoning about the more general task of agentic AI, specifically when we consider general agency as the act of dynamically constructing one's method of acting. Research into more sophisticated substantive reasoning mechanisms, and more general agentic architectures that embed such mechanisms, will need to occur if we wish to build automata capable of agentic behavior. Furthermore, these new agentic architectures will likely require enhanced meta-reasoning capacity, by being applied in far more complex environments than typical modern practical AI systems.

The agentic planning that we present in this work forms a specialized area of meta-reasoning that is of immediate practical interest. Over and above interesting theoretical questions about more general forms of meta-reasoning, there are also many additional questions to be considered in the context of agentic planning. For instance, the general form of agentic planning we describe here will likely require different algorithmic specifics to be viable in a variety of increasingly complex domains, ranging from simple to complex environments, or using different primitive goals or performance measures. The answers to these types of questions may be needed to inform the design of agentic planners within the AI Cognitive Architectures community seeking to build general-purpose cognitive systems. To give one such example – models have mostly relied on hand-coded procedural knowledge used for picking the best action, and that stands in contrast to the more general approach to agentic planning we outline here. As such, it may be valuable for the field of cognitive architectures, and beyond, to investigate the mechanism of action selection capability, and to see if a more general algorithm can be used to improve the system's learning capabilities.

Likewise, further concrete agentic meta-reasoning algorithms should be architected and tested away from hyper-planner, to ground-truth the research. We also give examples of algorithmic multitasking and scheduling domains in the work, and we note that those domains are quite lightweight compared to the real world. Overall, as increasing numbers of scheduling, multitasking, and planning algorithms become viable on practical timescales, the necessitation of grounding agentic planning becomes a reality.

References

1. Polineni, T. N. S., & Seenu, A. (2025). The New Frontier of Healthcare and Industry: Subash's Expertise in Big Data and Cloud Computing for Enhanced Operational Efficiency. Cuestiones de Fisioterapia, 54(2), 271-283.
2. Maguluri, K. K., Ganti, V. K. A. T., Yasmeen, Z., & Pandugula, C. (2025, January). Progressive GAN Framework for Realistic Chest X-Ray Synthesis and Data Augmentation. In 2025 6th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI) (pp. 755-760). IEEE.
3. Koppolu, H. K. R. Deep Learning and Agentic AI for Automated Payment Fraud Detection: Enhancing Merchant Services Through Predictive Intelligence.
4. Nampalli, R. C. R., & Adusupalli, B. (2024). AI-Driven Neural Networks for Real-Time Passenger Flow Optimization in High-Speed Rail Networks. Nanotechnology Perceptions, 334-348.
5. Chakilam, C. (2022). Generative AI-Driven Frameworks for Streamlining Patient Education and Treatment Logistics in Complex Healthcare Ecosystems. Kurdish Studies. Green Publication. https://doi. org/10.53555/ks. v10i2, 3719.
6. Sriram, H. K. (2023). Harnessing AI Neural Networks and Generative AI for Advanced Customer Engagement: Insights into Loyalty Programs, Marketing Automation, and Real-Time Analytics. Educational Administration: Theory and Practice, 29(4), 4361-4374.
7. Burugulla, J. K. R. (2025). Enhancing Credit and Charge Card Risk Assessment Through Generative AI and Big Data Analytics: A Novel Approach to Fraud Detection and Consumer Spending Patterns. Cuestiones de Fisioterapia, 54(4), 964-972.
8. Chava K. Dynamic Neural Architectures and AI-Augmented Platforms for Personalized Direct-to-Practitioner Healthcare Engagements. J Neonatal Surg [Internet]. 2025Feb.24 [cited 2025Mar.24];14(4S):501-10. Available from: https://www.jneonatalsurg.com/index.php/jns/article/view/1824
9. Challa, K. (2024). Neural Networks in Inclusive Financial Systems: Generative AI for Bridging the Gap Between Technology and Socioeconomic Equity. MSW Management Journal, 34(2), 749-763.
10. Sondinti, K., & Reddy, L. (2025). The Future of Customer Engagement in Retail Banking: Exploring the Potential of Augmented Reality and Immersive Technologies. Available at SSRN 5136025.
11. Malempati, M., & Rani, P. S. Autonomous AI Ecosystems for Seamless Digital Transactions: Exploring Neural Network-Enhanced Predictive Payment Models.

12. Pallav Kumar Kaulwar. (2023). Tax Optimization and Compliance in Global Business Operations: Analyzing the Challenges and Opportunities of International Taxation Policies and Transfer Pricing. International Journal of Finance (IJFIN) - ABDC Journal Quality List, 36(6), 150-181.
13. Vankayalapati, R. K. (2025). Architectural foundations of hybrid cloud. The Synergy Between Public and Private Clouds in Hybrid Infrastructure Models: Real-World Case Studies and Best Practices, 17.
14. Nuka, S. T. (2025). Leveraging AI and Generative AI for Medical Device Innovation: Enhancing Custom Product Development and Patient Specific Solutions. Journal of Neonatal Surgery, 14(4s).
15. Rao Suura S. Agentic AI Systems in Organ Health Management: Early Detection of Rejection in Transplant Patients. J Neonatal Surg [Internet]. 2025Feb.24 [cited 2025Mar.24];14(4S):490-50.
16. Kannan, S. (2025). Transforming Community Engagement with Generative AI: Harnessing Machine Learning and Neural Networks for Hunger Alleviation and Global Food Security. Cuestiones de Fisioterapia, 54(4), 953-963.
17. Srinivas Kalisetty, D. A. S. Leveraging Artificial Intelligence and Machine Learning for Predictive Bid Analysis in Supply Chain Management: A Data-Driven Approach to Optimize Procurement Strategies.
18. Challa, S. R. Diversification in Investment Portfolios: Evaluating the Performance of Mutual Funds, ETFs, and Fixed Income Securities in Volatile Markets.
19. Vamsee Pamisetty. (2023). Intelligent Financial Governance: The Role of AI and Machine Learning in Enhancing Fiscal Impact Analysis and Budget Forecasting for Government Entities. Journal for ReAttach Therapy and Developmental Diversities, 6(10s(2), 1785–1796. https://doi.org/10.53555/jrtdd.v6i10s(2).3480
20. Komaragiri, V. B. (2022). AI-Driven Maintenance Algorithms For Intelligent Network Systems: Leveraging Neural Networks To Predict And Optimize Performance In Dynamic Environments. Migration Letters, 19, 1949-1964.
21. Annapareddy, V. N., & Rani, P. S. AI and ML Applications in RealTime Energy Monitoring and Optimization for Residential Solar Power Systems.
s