Is Artificial Intelligence the "new" Operating System?

Omer F. Rana

School of Computer Science & Informatics

Cardiff University, UK

ranaof@cardiff.ac.uk Twitter:@omerfrana

CARDIFF UNIVERSITY

PRIFYSGOL CAERDYDD

This talk is really about initiating a discussion …
and identify potential research directions

If Artificial Intelligence techniques are embedded in our systems and software environments – how should a developer respond to this?

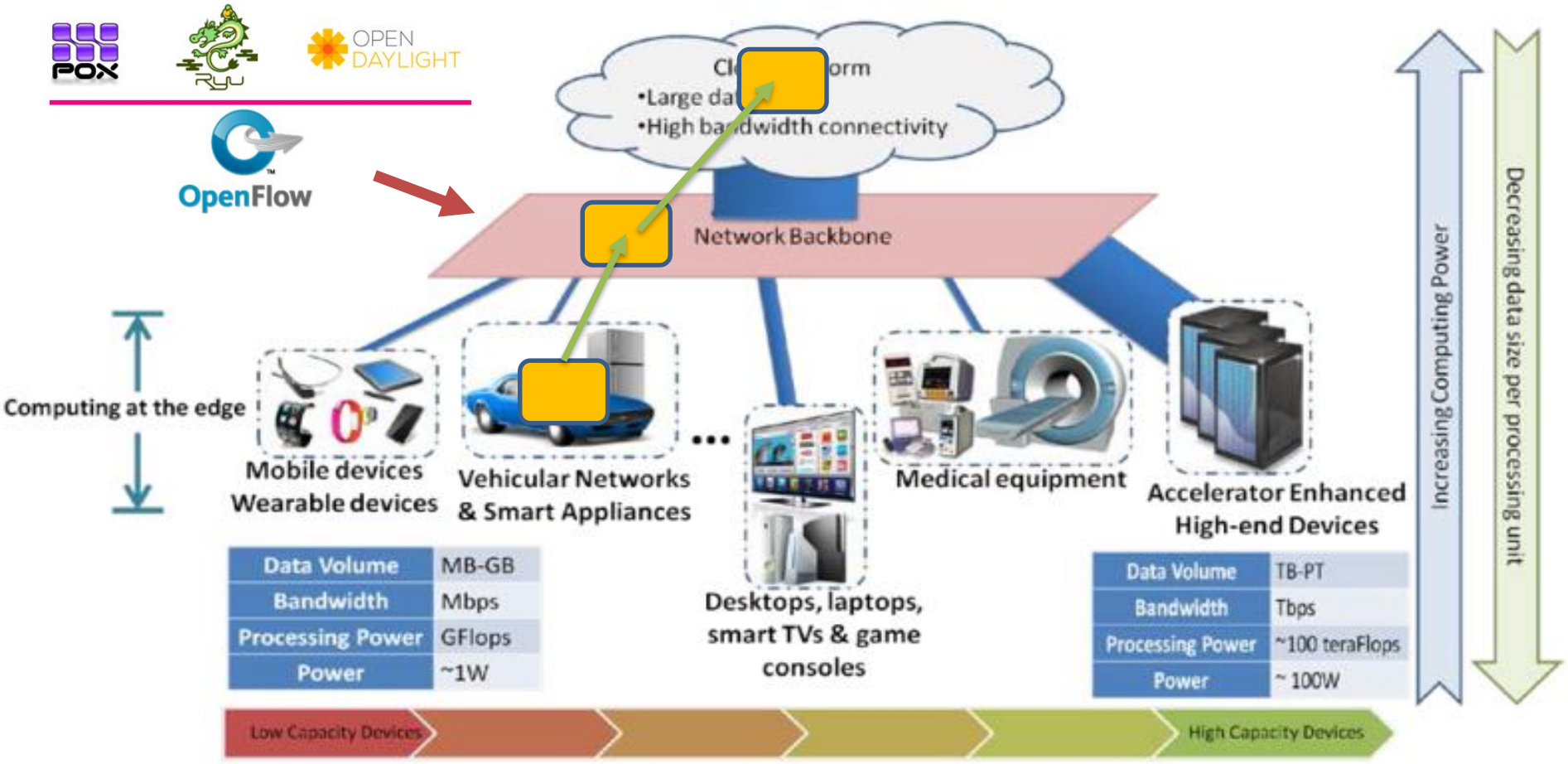
Re-create, Re-use, Adapt or Ignore?

Should we differentiate between Artificial Intelligence as an enabler
vs. Artificial Intelligence as an application?

To what extent should we “trust” the outcome of Artificial Intelligence-based systems?
How do we decide?

Edge & Programmability
(from Manish Parashar)

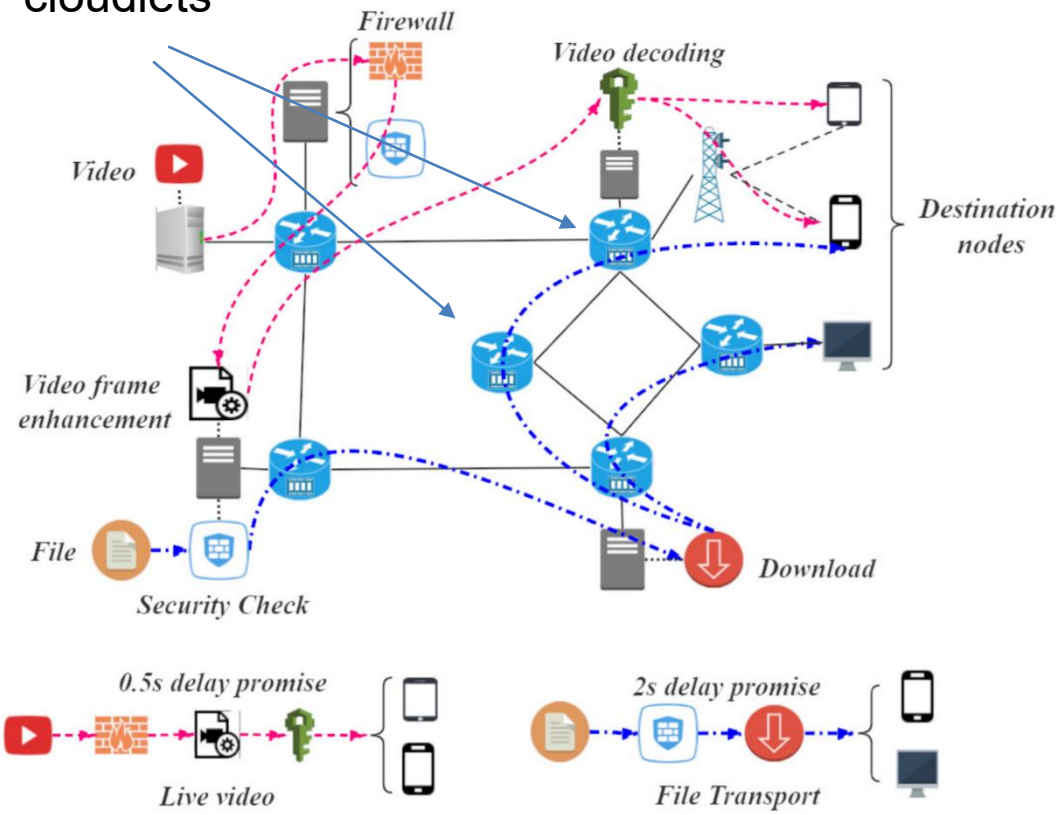
Software Defined Networks

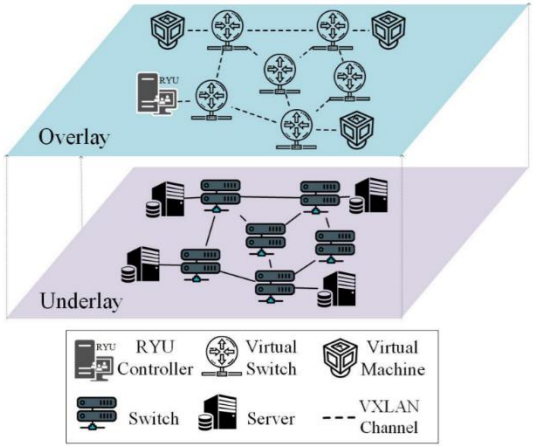


Software Defined Networking enables telecom. operators to offer specialist data services. Computation extends beyond the Data Centre … enabling additional capability & revenue streams. **Merging of Data Centre & Telecom. functionality**

Edge & Intransit capability
(AI to optimise infrastructure & placement)

“cloudlets”





(a) The underlay and overlay of the test-bed

(b) The physical deployment of the hardware switches

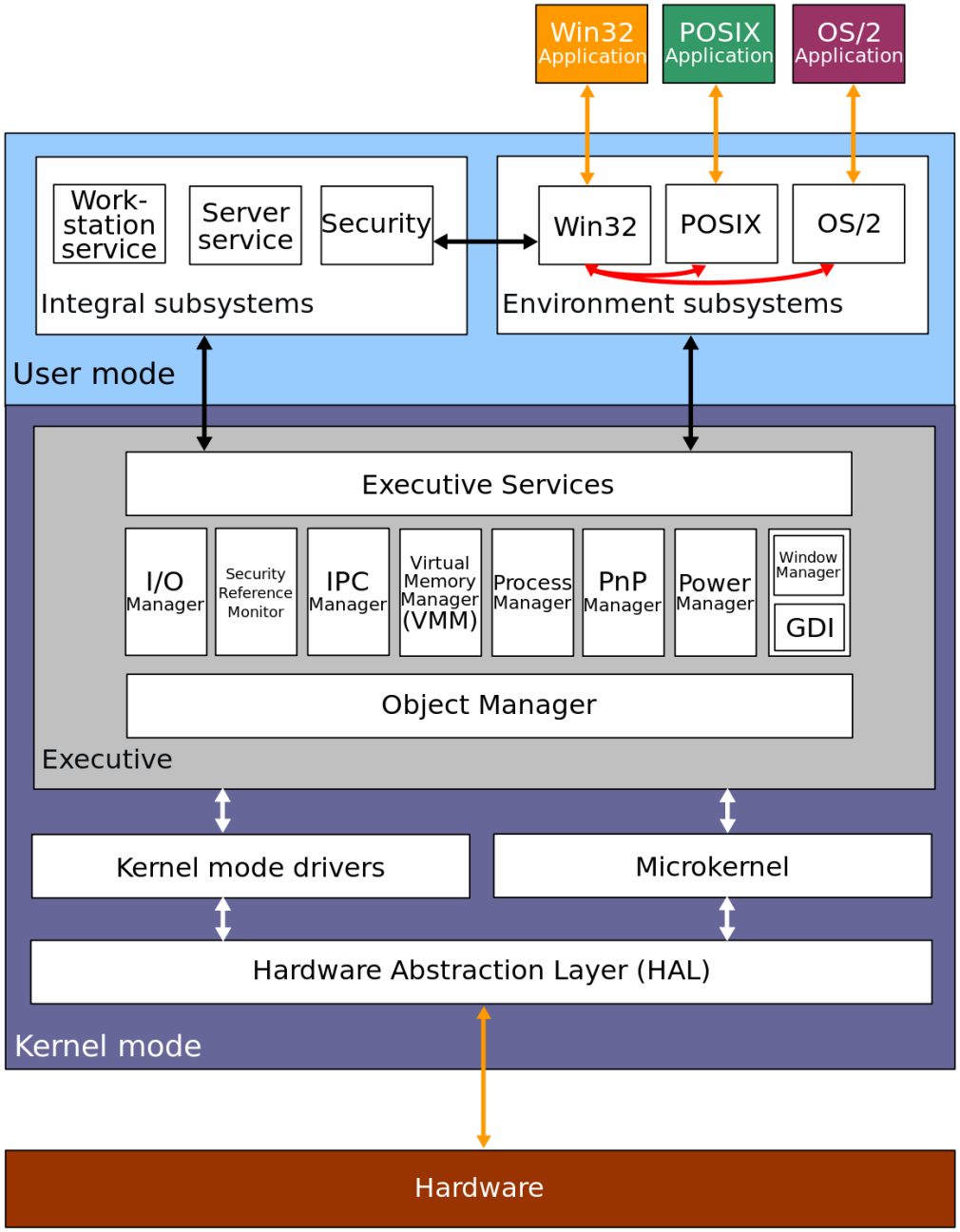
(c) The raspberry pi

constraints of cloudlets.

Qin, Yugen, Xia, Qiufen, Xu, Zichuan, Zhou, Pan, Galis, Alex, Rana, Omer, Ren, Jiankang and Wu, Guowei, “Enabling multicast slices in edge networks”. *IEEE Internet of Things* 7 (9) , 2020, pp. 8485-8501.

What is an Operating System?

- Managing computational resources to meet particular Quality of Service criteria
  - Process/Thread management & Coordination
  - Kernel architectures
  - Resource management: memory, I/O, resources (internal and external), network
  - Managing user migration and state
  - Extension: Plug-ins, Dynamic Link Libraries ++
  - Security support (at various levels: SSL/TLS, Access Control ++)
- Abstraction driven (process or threads)
- All the software you did not write, but must "trust"



User application interface

Support for coordination/ management Services

Abstraction support

User vs. Kernel Model

Support for extending capability

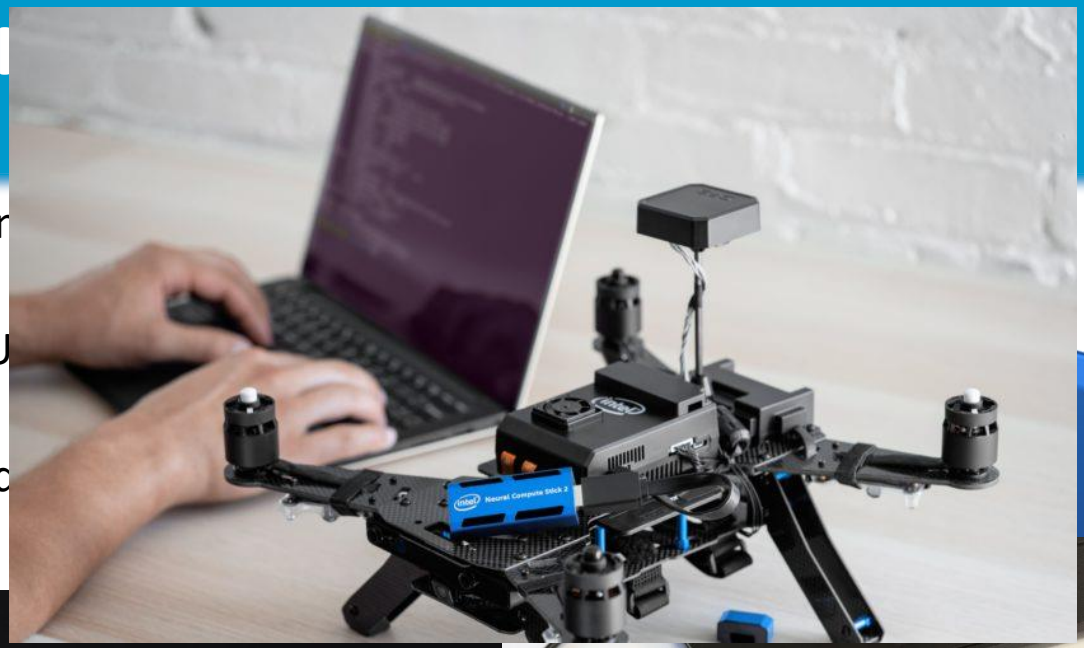
Many variations on this basic model – including networked and embedded OS

Edge Environments: I

Deep neural network (DNN) inferer
the chip – 16 SHAVE cores

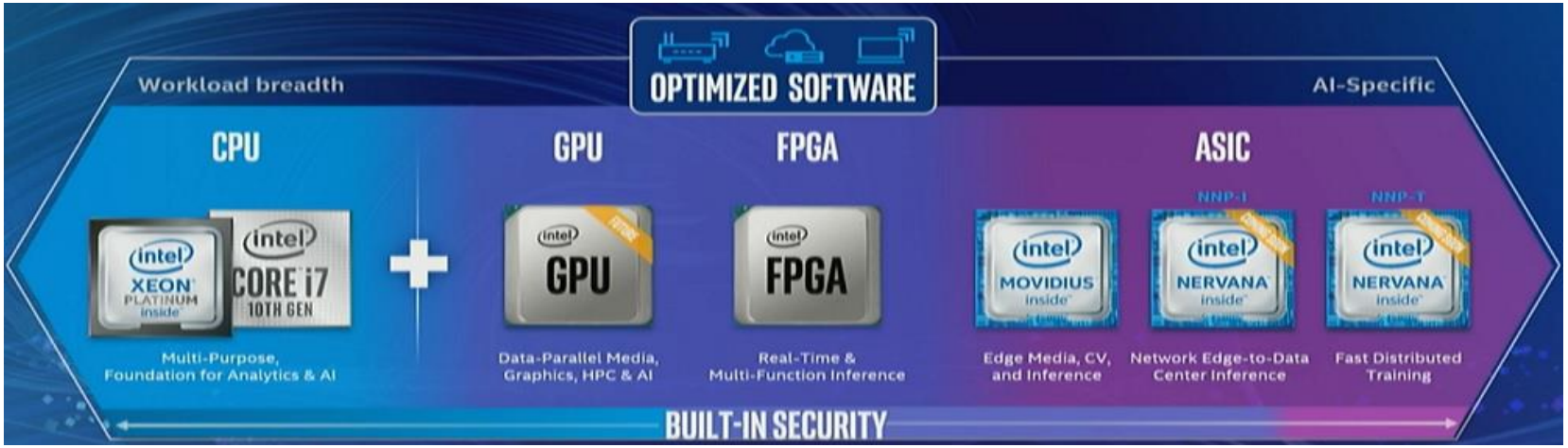
Prototype on stick – deploy on VPU
embedded device(s)

Intel OpenVINO toolkit -- used for d



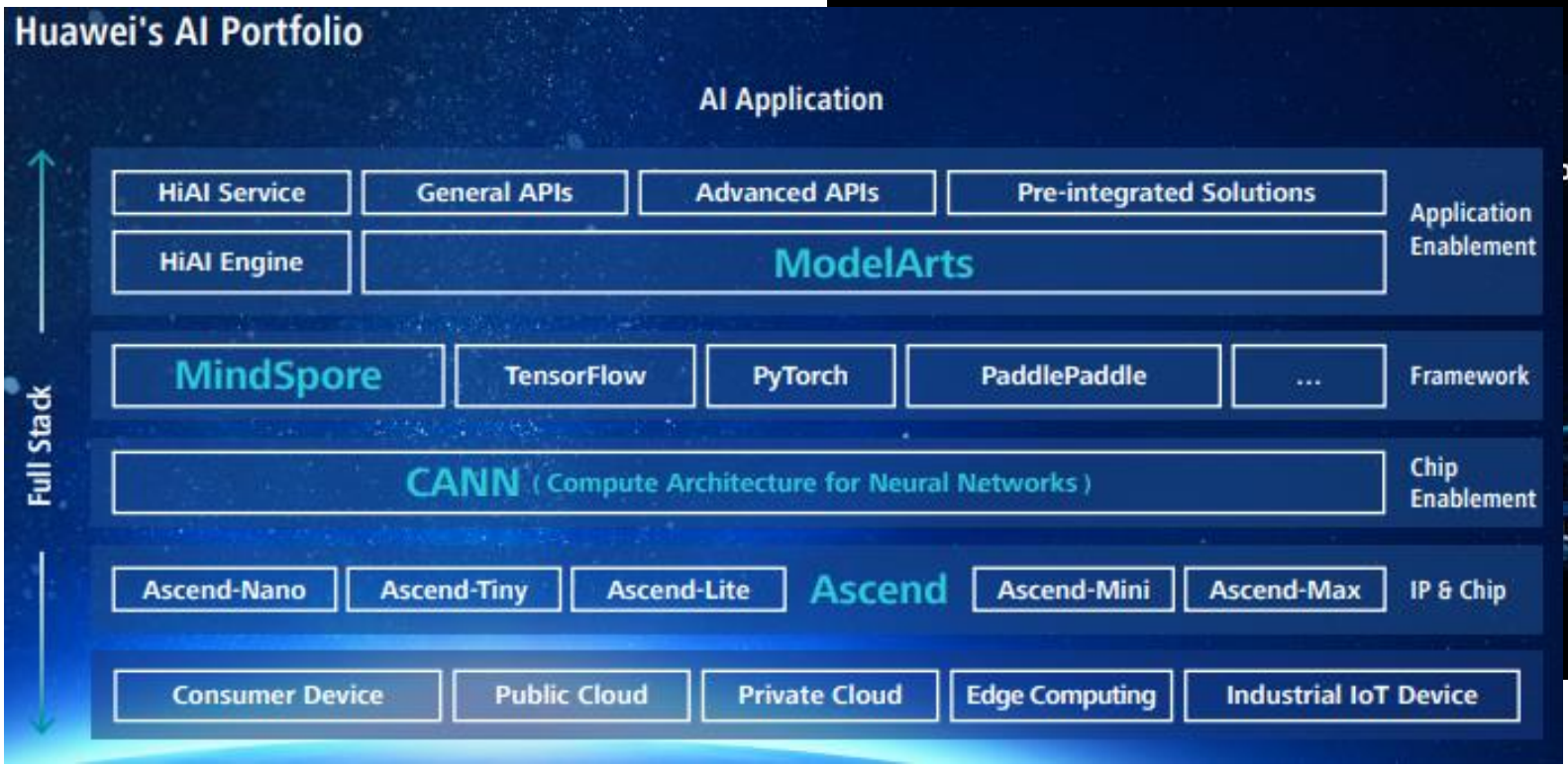
Edge Environments: Intel Vision Processing Units (VPUs) -- https://www.intel.ai/featured-solutions/

- VPUs feature purpose-built (SHAVE) processor cores – for speeding up neural net. execution & programmed via Myriad Development Kit (MDK).
- VPUs in Google's Clips camera, DJI's Phantom 4 drone, and Tencent DeepGaz (China) ++



AI-based enhancements

Hardware vendors support AI:

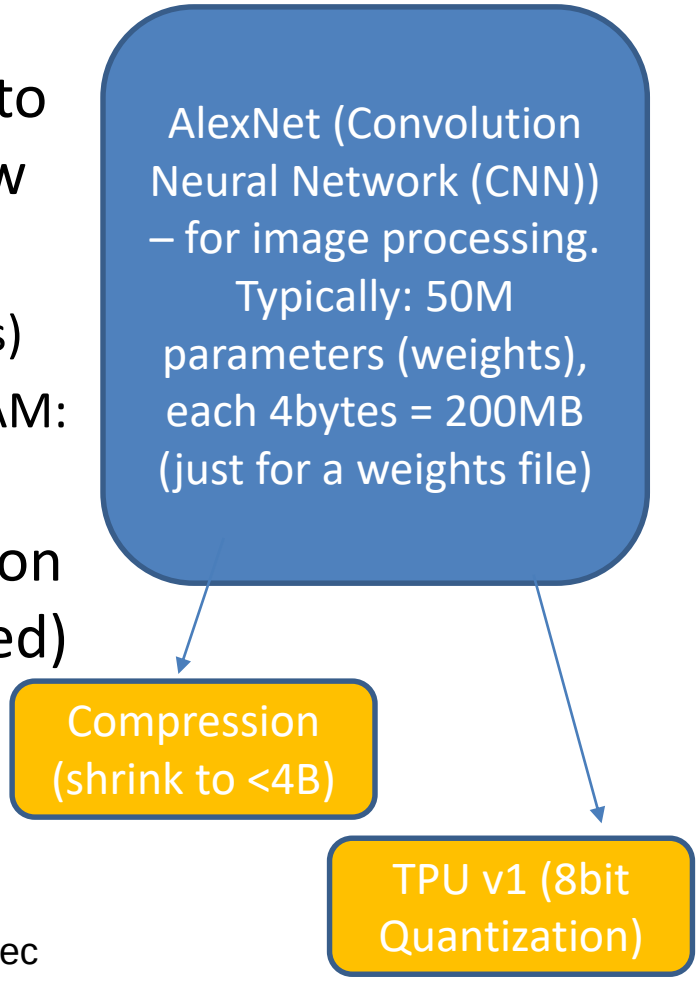


Hardware & Systems vendors (Intel, Dell, NVidia, ++) are embedded AI into their Platform (Huawei Da Vinci architecture)

TinyML – Machine Learning for Edge Environments

https://www.tinyml.org/summit/

- Reducing size of learned model (e.g. MobileNet(v2) + SSDLite) – and migrating to edge resources (Arduino) – e.g. TensorFlow Lite
  - Bandwidth, Latency, Security (some key drivers)
  - Limited device memory (Flash: 32KB-~MB; SRAM: 16KB – 512KB)
- Training on a Cloud environment: Distillation & Quantization of a learned model (C-based)
- Trading off accuracy vs. performance



https://www.tensorflow.org/lite/performance/quantization_spec

PerfML for TinyML

https://groups.google.com/g/mlperf-tiny

Type of processors:

AMD Cortex A7, STM32 Microcontroller, Raspberry Pi, Arduino Uno

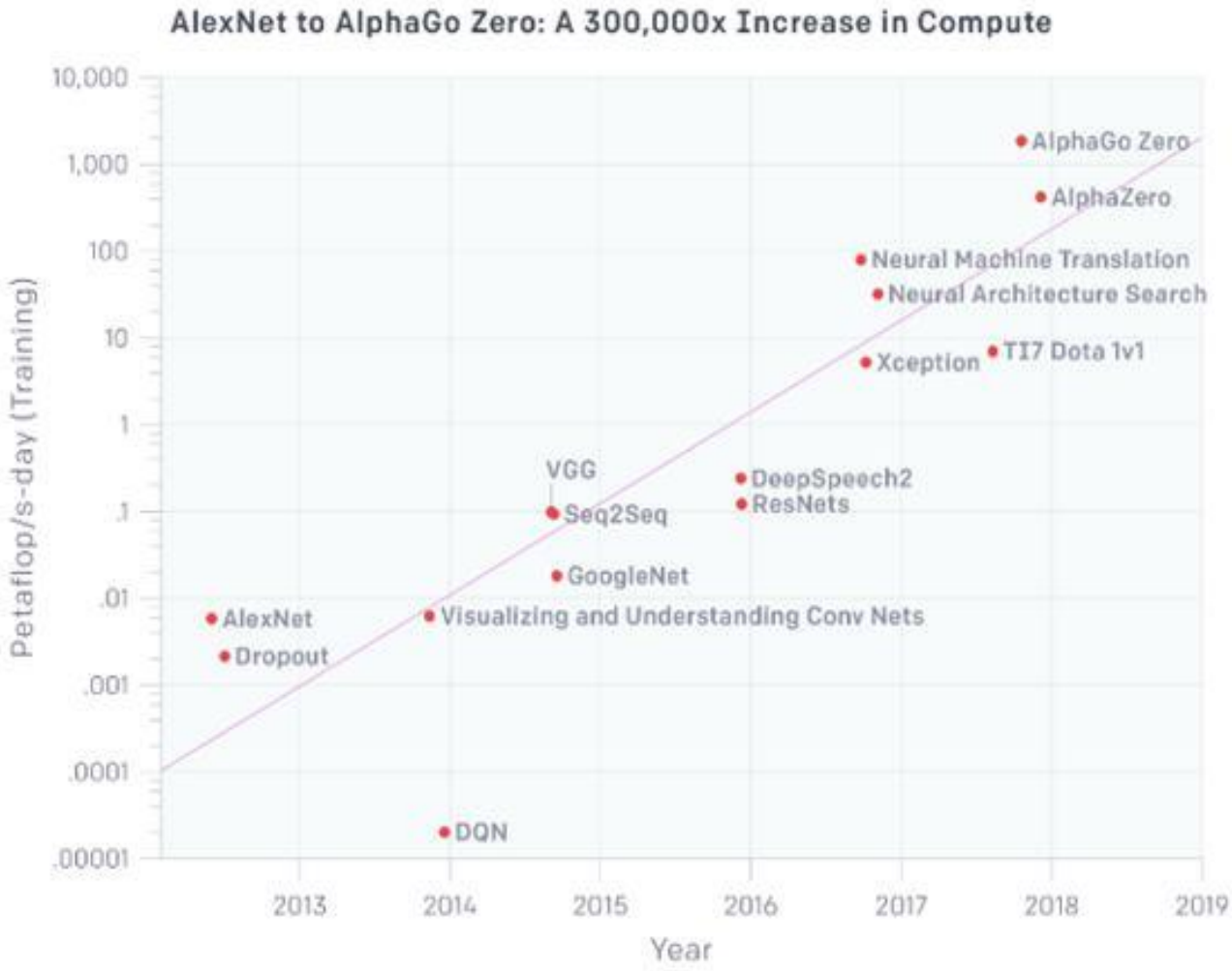
Task Category	Use Case	Model Type	Datasets
Audio	Audio Wake Words Context Recognition Control Words Keyword Detection	DNN CNN RNN LSTM	Speech Commands Audioset ExtraSensory Freesound DCASE
Image	Visual Wake Words Object Detection Gesture Recognition Object Counting Text Recognition	DNN CNN SVM Decision Tree KNN Linear	Visual Wake Words CIFAR10 MNIST ImageNet DVS128 Gesture
Physiological / Behavioral Metrics	Segmentation Anomaly Detection Forecasting Activity Detection	DNN Decision Tree SVM Linear	Physionet HAR DSA Opportunity
Industry Telemetry	Sensing Predictive Maintenance Motor Control	DNN Decision Tree SVM Linear Naive Bayes	UCI Air Quality UCI Gas UCI EMG NASA's PCoE

TensorFlow Lite

https://www.tensorflow.org/lite/

- **Interpreter** for on-device "inference", supporting a set of core operators that are adapted for specific devices (with a small binary size)
  - interpreter **uses a static graph ordering** and a custom (less-dynamic) memory allocator
  - a **converter** that maps TensorFlow model (including Keras) → TensorFlow Lite
- Includes Model optimization tools, including quantization, that can reduce size and increase performance of models without sacrificing accuracy.
- Uses **FlatBuffer** (https://google.github.io/flatbuffers/) that is optimized for small size and portability (serialized access, no packing/unpacking)
- Supports Android & iOS, embedded Linux, and microcontrollers (Arduino)
  - API support for Java, Swift, Objective-C, C++, and Python
- Can work with hardware acceleration on supported devices (e.g. quantized → float if GPU is available), device-optimized kernels
- **Pre-trained models** *(https://www.tensorflow.org/lite/models)* for common machine learning tasks (image analysis, object detection, pose estimation, image styling/transfer, NLP ++) – hosted models: https://www.tensorflow.org/lite/guide/hosted_models

Compute Support for AI



Compute used in the largest AI training runs has grown >300,000x.

A petaflop/s-day (pfs-day) consists of performing 1015 neural net operations per second for one day

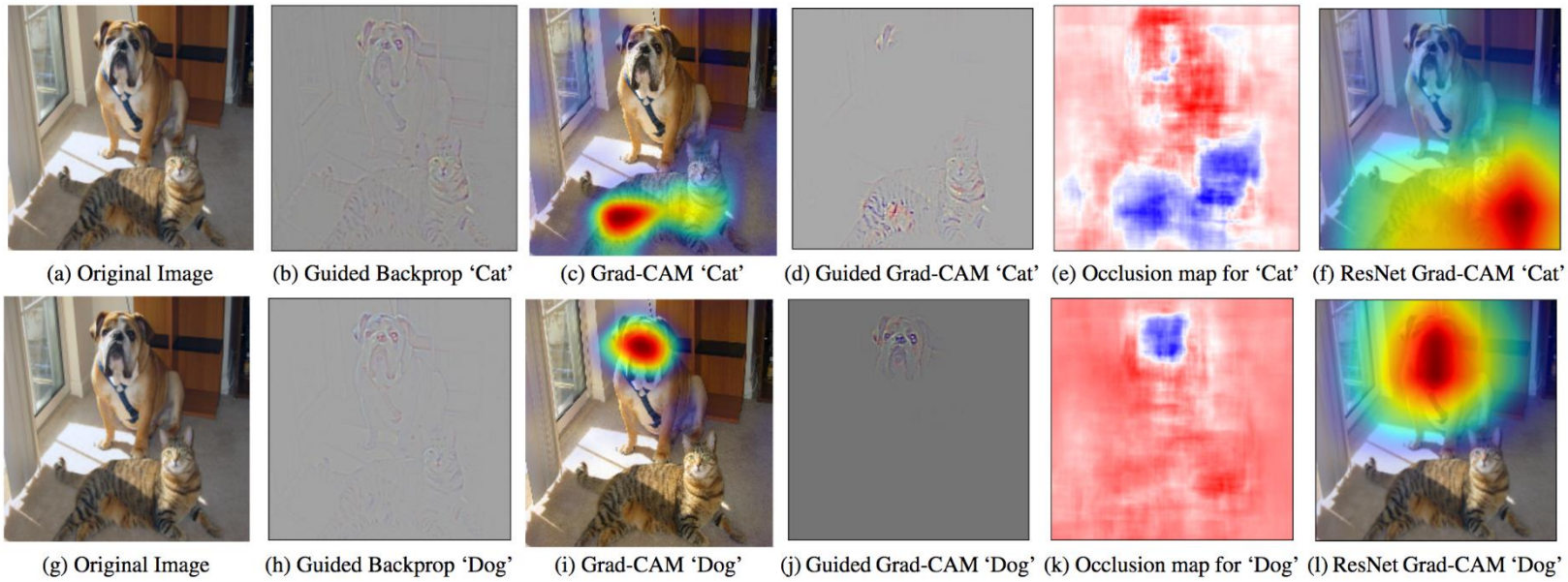
https://www.i-programmer.info/news/105-artificial-intelligence/11823-an-exponential-law-for-ai-compute.html

Do we understand what the AI does?

http://www.shallowmind.co/jekyll/pixyll/2017/12/30/tree-regularization/

- Lots of focus on “explainable AI”
- Balancing interpretability with accuracy
  - Use of approximations is key to support this
  - But not always possible
- Identify “focus” of a deep learning network – generating heat map on images

GradCam - Creating visual explanations for decisions by using gradients of target concepts to highlight important pixels



Do we understand what the AI does? “Feature Visualisation”

Dataset Examples show us what neurons respond to in practice

Optimization isolates the causes of behavior from mere correlations. A neuron may not be detecting what you initially thought.

Baseball—or stripes?
mixed4a, Unit 6

Animal faces—or snouts?
mixed4a, Unit 240

Clouds—or fluffiness?
mixed4a, Unit 453

Buildings—or sky?
mixed4a, Unit 492

- Feature Visualization - generate images via optimization to activate a specific neuron or set of neurons
- Isolating effects of individual neurons

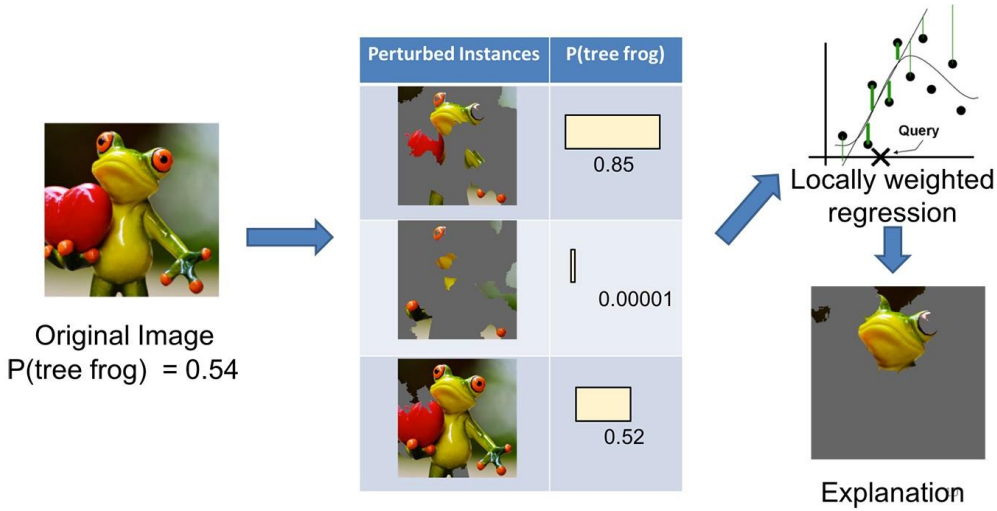
Generating Explanations

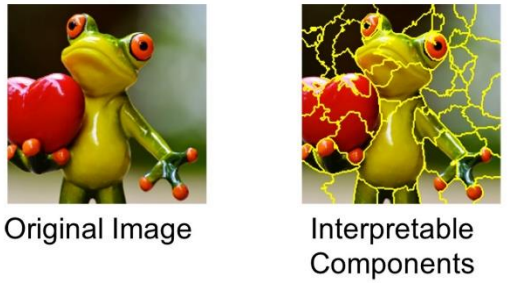
- "Contrastive Explanation Method" (IBM)
  - Generates instance based local black box explanations for classification models in terms of Pertinent Positives (PP) and Pertinent Negatives (PN)
  - PP: features that should be minimally and sufficiently present (e.g. important pixels in an image) to predict the same class as on the original instance
  - PN: features should be minimally and necessarily absent from the instance to be explained in order to maintain the original prediction class

LIME – locally explainable models

https://towardsdatascience.com/understanding-how-lime-explains-predictions-d404e5d1829c

- Local Interpretable Model-Agnostic Explanations (LIME)
  - even the most complex model can be reasonably approximated locally using a simpler model
- LIME generates an explanation by approximating the underlying model by an interpretable one (such as a linear model)





Dynamic VM allocation & Elastic provisioning

- “Elasticity”

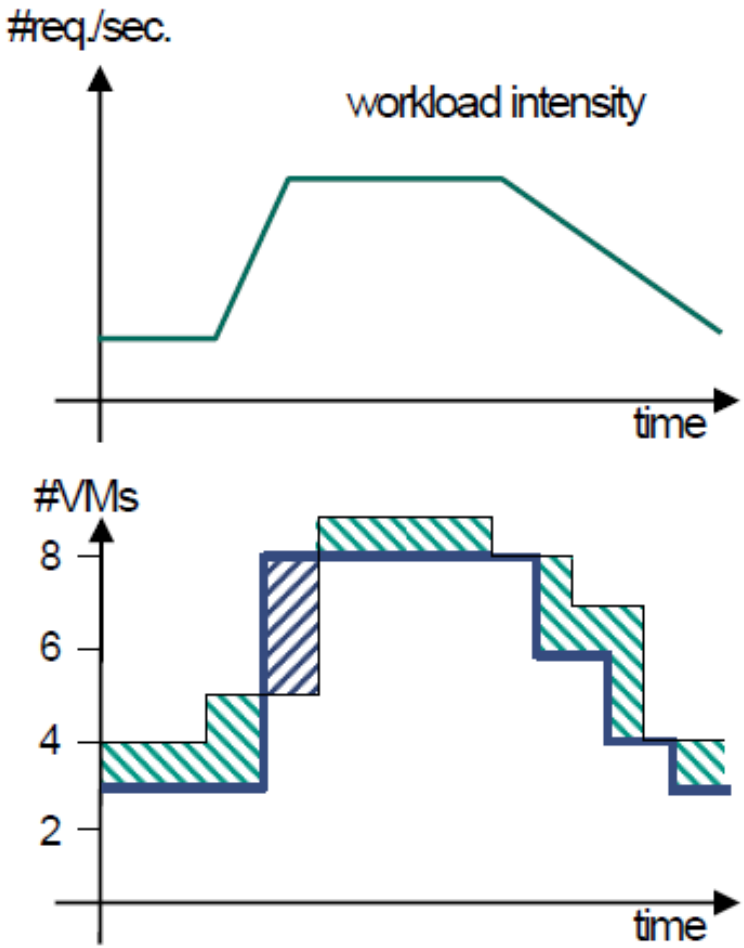
  the degree to which a system is able to **adapt** to **workload changes** by **provisioning** and **de-provisioning** resources, such that at each point in time the **available resources match** the **current demand** as closely as possible.

- Can elastic provisioning capability be measured?

Dynamic VM/container allocation

- Scale up speed: **switch** from an under provisioned state **to an optimal or overprovisioned state**.
  - Can we consider “temporal” aspects of how scaling up takes place
- Deviation from actual to required resource demand
  - Measure deviation to influence the overall process
- Role of predictive allocation for “known” events
  - i.e. know in advance how many VMs or container instances to allocate

An Example: Quality Metrics to drive AI



Service Level Agreement (SLA):

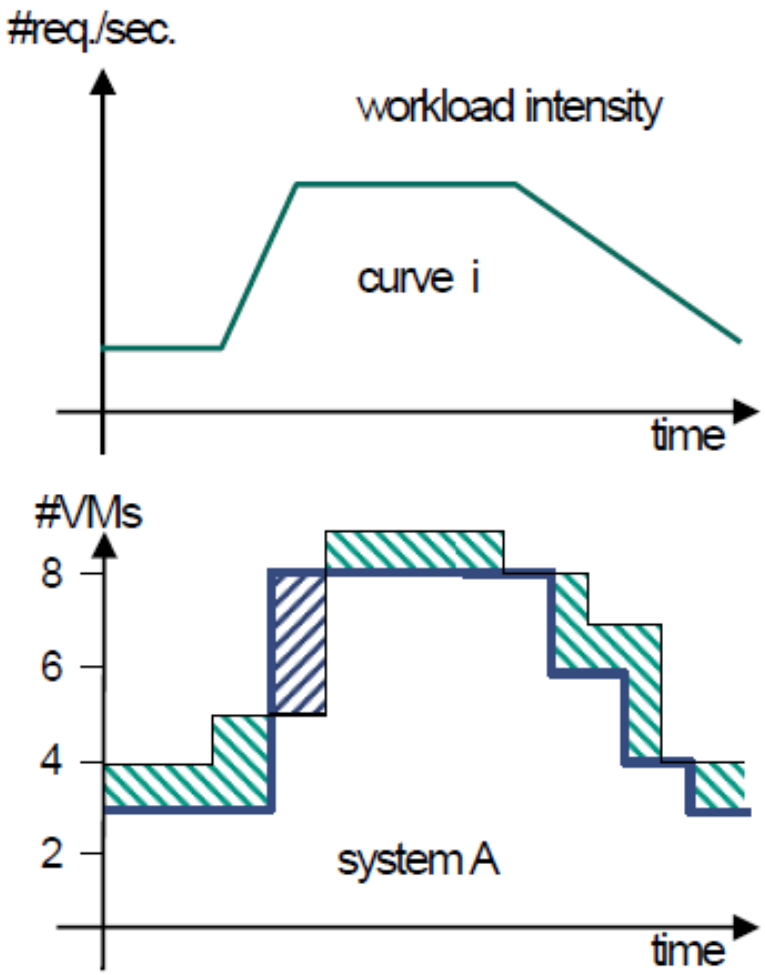
E.g.: resp. time ≤ 2 sec, 95%

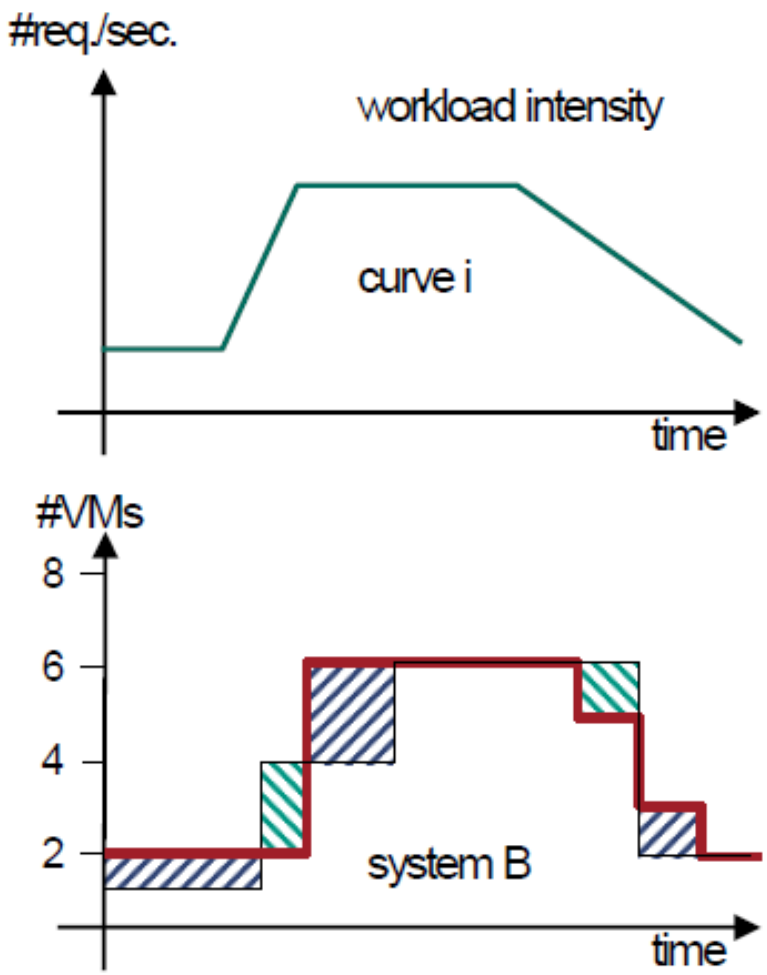
Resource Demand:

Minimal amount of #VMs required to ensure SLAs.

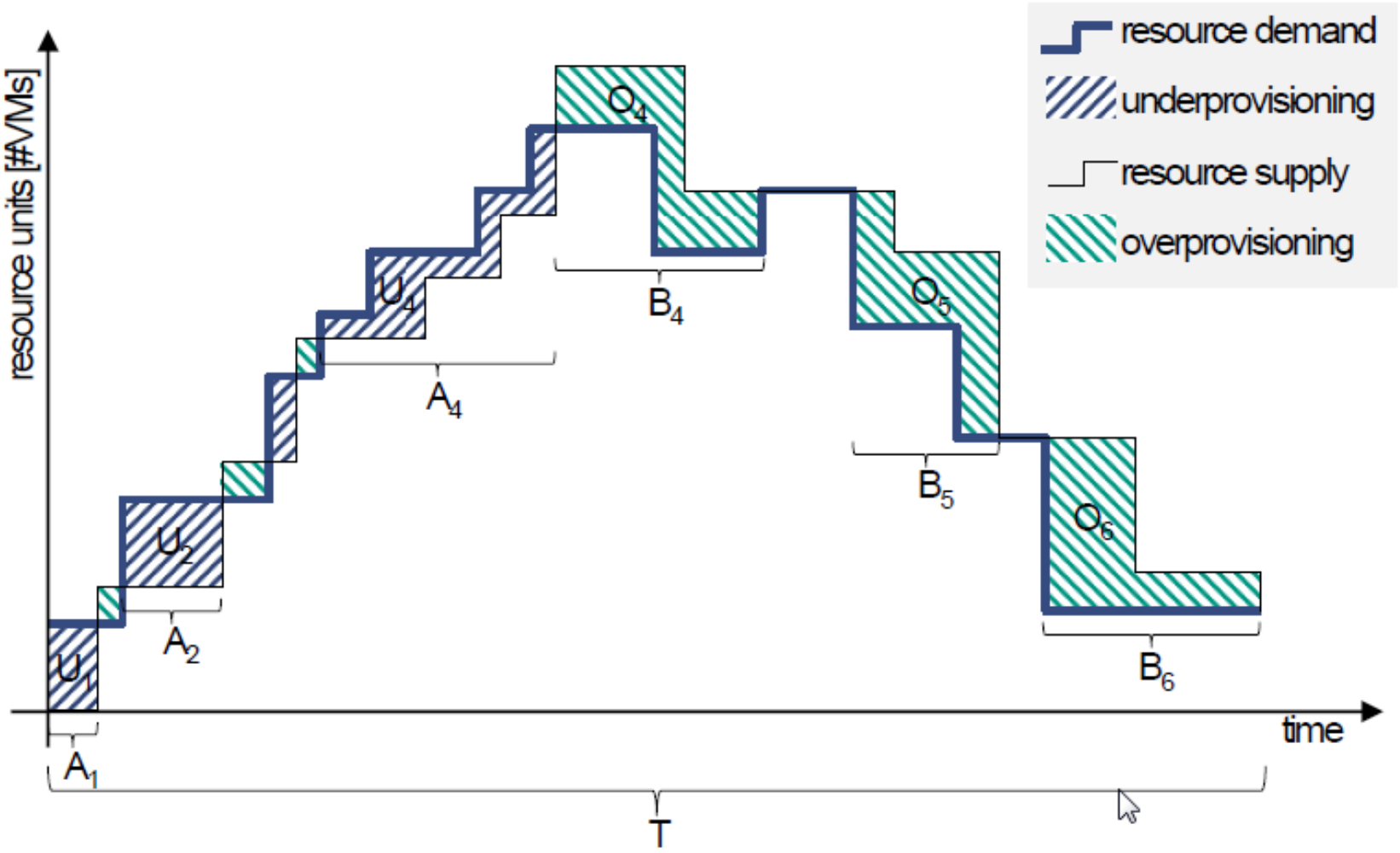
"**Elasticity in Cloud Computing: What It Is, and What It Is Not**"
Nikolas Herbst, Samuel Kounev, Ralf Reussner, ICAC 2013 (USENIX)

Comparing allocation





Elasticity Metrics



Elasticity Metrics … 2

$\bar{A}$	Average time of switch from an underprovisioned to an optimal or overprovisioned state II_rahmen_neu_folge **average speed of scaling up**
$\sum A$	Accumulated time in underprovisioned state.
$\bar{U}$	Average amount of underprovisioned resources during an underprovisioned period.
$\sum U$	Accumulated amount of underprovisioned resources.
$\bar{B}, \sum B, \bar{O}, \sum O$	correspondingly for overprovisioned states

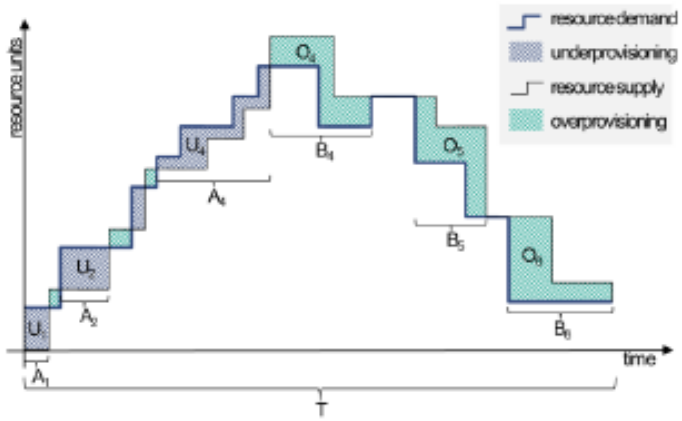
$$P_u = \frac{\sum U}{T}; P_d = \frac{\sum O}{T},$$

Average precision of scaling up / down

$T = total\ evaluation\ duration$

$$E_u = \frac{1}{\bar{A} x \bar{U}}; E_d = \frac{1}{\bar{B} x \bar{O}}$$

Elasticity metric for scaling up / down

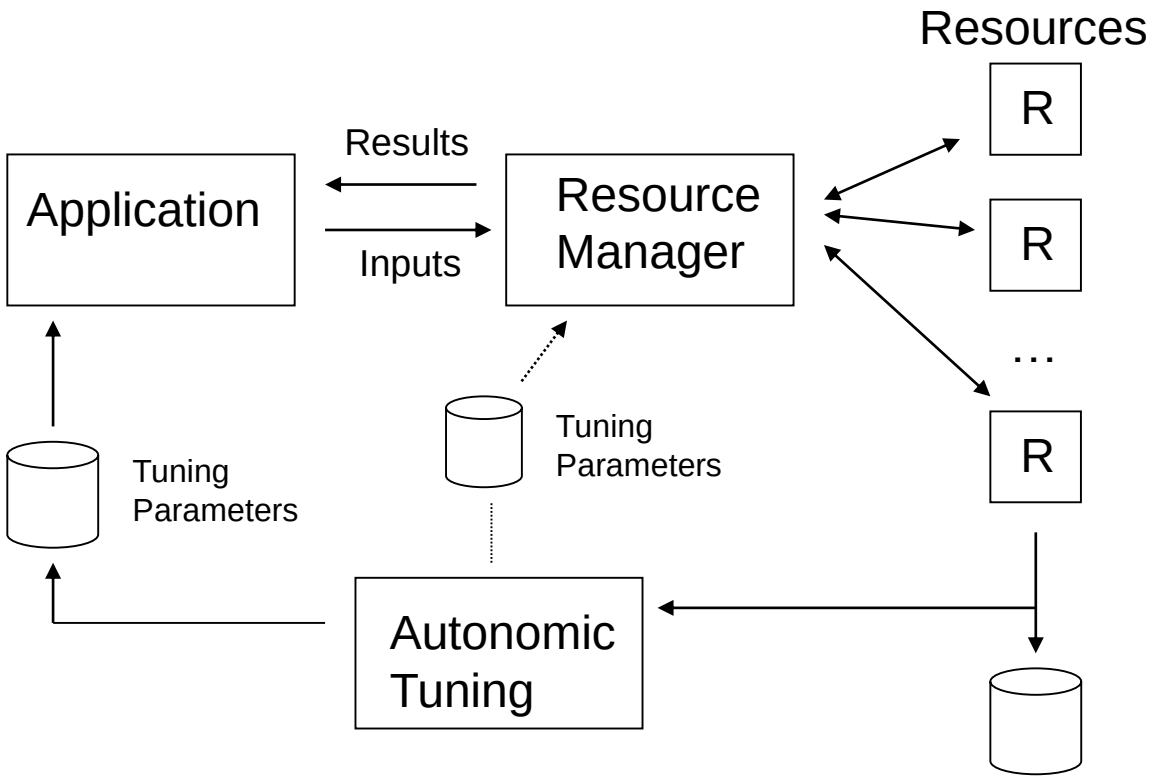


Developing Distributed Applications (DAs)

https://onlinelibrary.wiley.com/doi/full/10.1002/cpe.2897

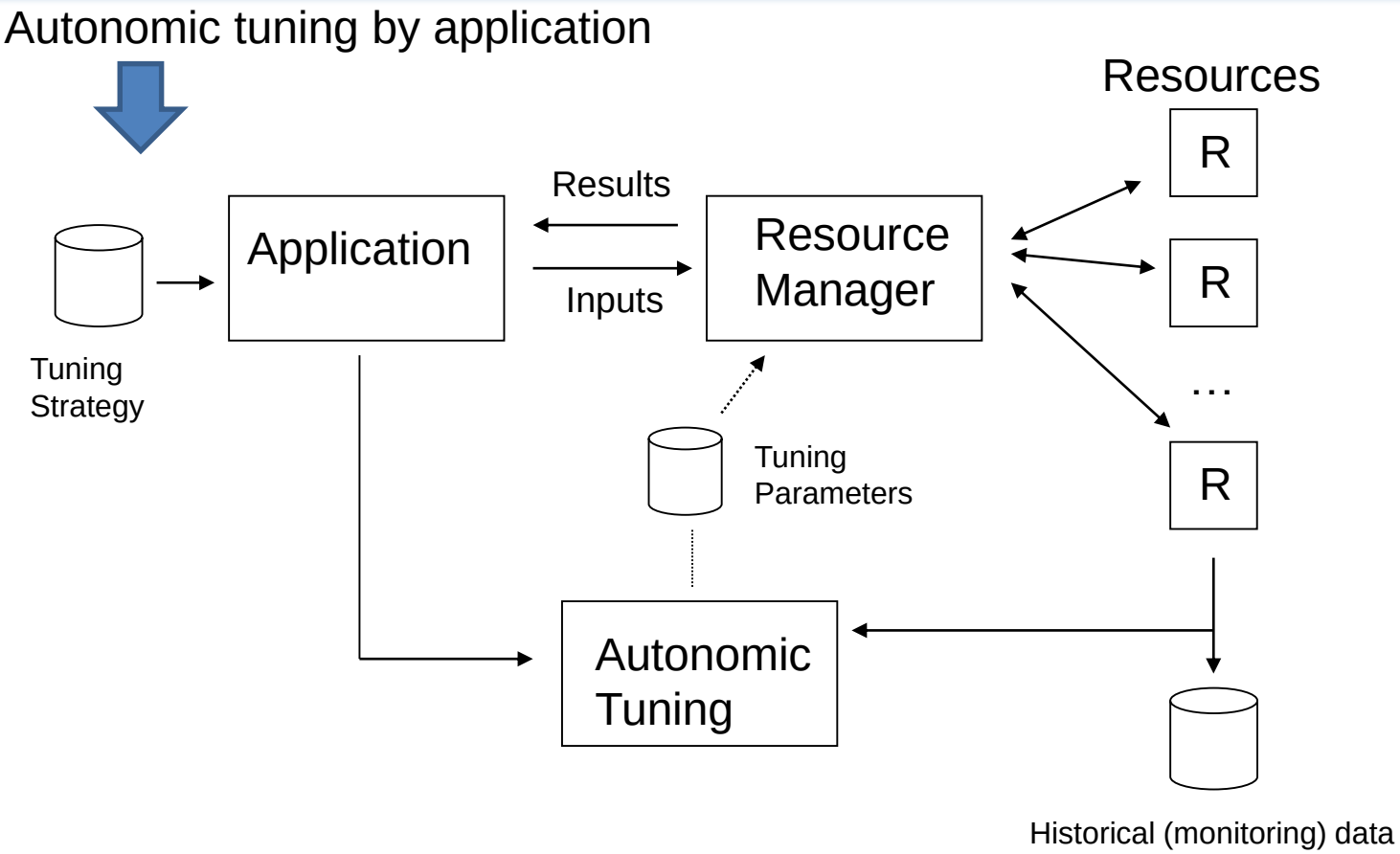
- How can applications make use of AI
  - Understanding what parameters are exposed to (one or more) external controller(s)
- Location/influence of the data controller
  - In practice, varying degree of control
- Separate control logic/behaviour from:
  - Data flow / path
  - Components engaged in the data path

Tuning of application & resource manager parameters



Autonomic **tuning of** application & resource manager parameters

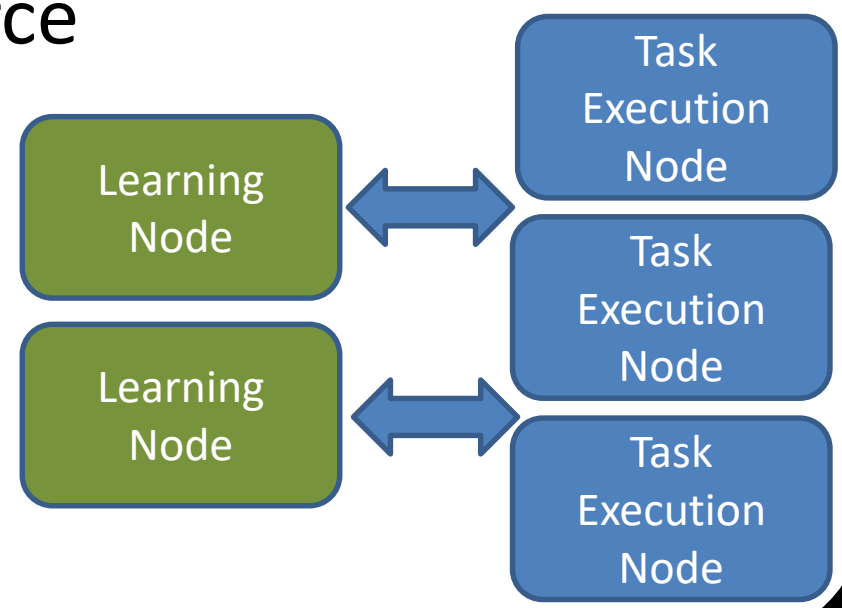
Tuning by application



- Resource reservation to achieve particular QoS-criteria
- Dynamic analysis of data stream from a scientific instrument – may also involve analysis of video/audio feeds

Splitting the resource pool (dynamically allocated)

- Combine machines in a resource pool
  - Some used to support simulation (actual workload)
  - Others used to tune the simulation ("learner" nodes)
- Dynamically adapt resource pool based on tuning



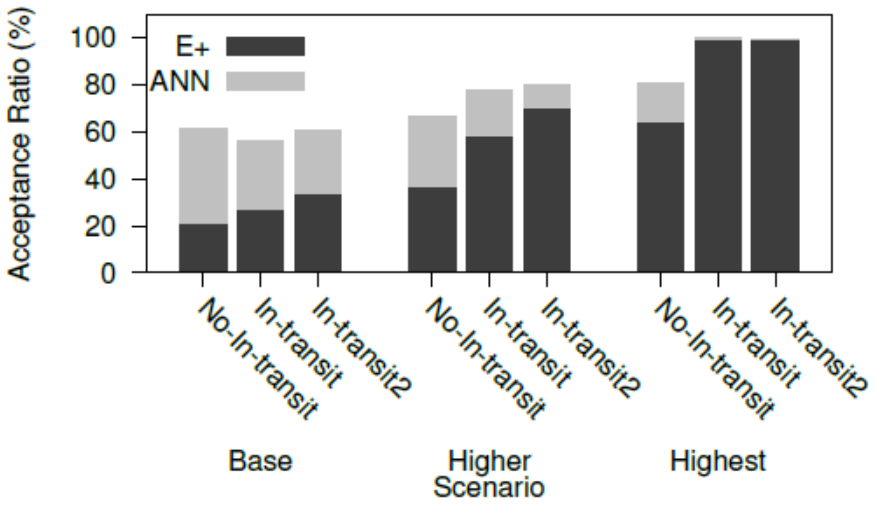
A. Zamani, M. Zou, J. Diaz-Montes, I. Petri, O. Rana, and M. Parashar, "A computational model to support in-network data analysis in federated ecosystems". *Future Generation Computer Systems* 80, 2018 -- pp. 342-354. 10.1016/j.future.2017.05.032

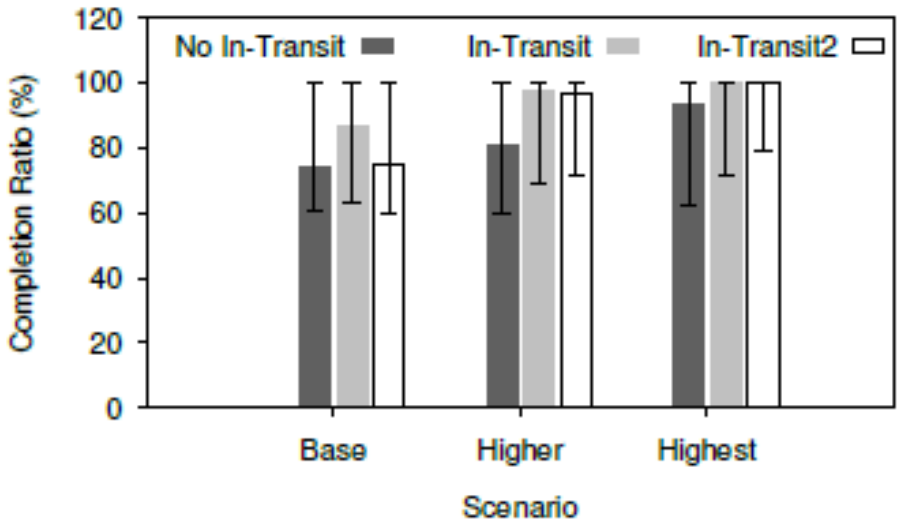
Edge Approximation …

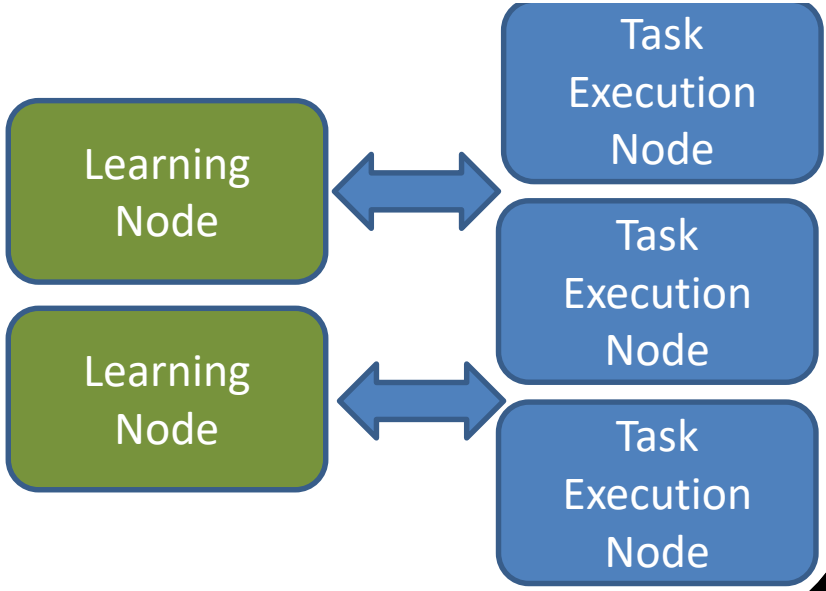
- Combine capability in Data Centre with “approximate” algorithms in transit or at the edge
- EnergyPlus (as at present) + a trained neural network (as a function approximator for EnergyPlus behaviour)
- But why?
  - EnergyPlus ~ Execution time(Minutes)
  - Neural Network Training ~ Execution time (Minutes)
  - Trained (FF) Neural Network ~ Execution time (Seconds)
- Combine more accurate model execution with approximate model via a learned neural network
- Trigger re-training when input parameters change significantly
  - Each EnergyPlus execution provides potential training data for the neural network

A. R. Zamani, I. Petri, J. Diaz-Montes, O. Rana, M. Parashar, ***“Edge-supported Approximate Analysis for Long Running Computations”,*** *Proceedings of the 5th IEEE International Conference on Future Internet of Things and Cloud (FiCloud 2017), Prague, Czech Rep., August 2017*

Splitting the resource pool
(dynamically allocated)







A. Zamani, M. Zou, J. Diaz-Montes, I. Petri, O. Rana, and M. Parashar, “A computational model to support in-network data analysis in federated ecosystems”. *Future Generation Computer Systems* 80 , pp. 342-354. 10.1016/j.future.2017.05.032

Distributed, Dynamic, Data Intensive (D3) Science : Application Survey

- Consider applications where data workload is equivalent to or greater than computational workload

- Properties of "dynamic" data applications
  - Real time: generated, re-distributed or partitioned data
  - On-demand: varying availability of data
  - Adaptive: change in processing or storage granularity

- Computational activities triggered due to data creation
  - Computation must respond to unplanned changes in data volumes or content

- To scale – various trade-offs necessary:
  - Move computation or data?
  - Frequency of access & varying modes of analysis
  - Increasing "in-memory" processing (high disk I/O overhead)

Considered a variety of applications that conform to the above characteristics

Distributed, Dynamic, Data Intensive (D3) Science : Application Survey (joint EPSRC (UK) & NSF (US) funded)

- What is the purpose of the application?
- How is the application used to do this?
- What infrastructure is used?
  - including compute, data, network, instruments, etc.
- What dynamic data is used in the application?
  - What are the types of data?
  - What is the size of the data set(s)?
- How does the application get the data?
- What are the time (or quality) constraints on the application?
- How much diverse data integration is involved?
- How diverse is the data?

Introducing Distributed Dynamic Data-intensive (D3) Science: Understanding Applications and Infrastructure, Shantenu Jha, Neil Chue Hong, Daniel S. Katz, Andre Luckow, Omer Rana, Yogesh Simmhan, Concurrency and Computation: Practice & Experience, John Wiley, 2017

https://arxiv.org/abs/1609.03647

Scientific Applications considered …

Application example	execution unit	Communication (data exchange)	Coordination	Execution environment
Montage	Multiple sequential and parallel executables	Files	Dataflow (DAG)	Dynamic process creation, workflow execution, file transfer
NEKTAR	Multiple concurrent instances of single executable	Messages	SPMD	MPI, coscheduling
Coupled fusion simulation	Multiple concurrent parallel executables	Stream-based	Dataflow	Coscheduling, data streaming, async. data I/O
Asynchronous replica-exchange	Multiple sequential and/or parallel executables	Pub/sub	Dataflow and events	Decoupled coordination and messaging, dynamic task generation
ClimatePrediction.net (generation)	Multiple sequential executables, distributed data stores	Files and messages	Master/ worker, events	At-Home (BOINC)
ClimatePrediction.net (analysis)	A sequential executable, multiple sequential or or parallel executables	Files and messages	Dataflow (Forest)	Dynamic process creation, workflow execution
SCOOP	Multiple different parallel executables	Files and messages	Dataflow	Preemptive scheduling, reservations

What does the AI manage?

- Distributed Application (DA) Vectors:
  - What the pieces of distribution are? How these pieces interact? Flow of information? What is needed to actually deploy and execute the application?
- DA Vectors: Preliminary theoretical framework to analyze DA structure
- Vectors: Axes representing application characteristics; understanding the value helps:
  - App Requirements
  - Skillful aggregation versus Decomposition
    - Primacy of coordination
  - Design, constraints of solutions

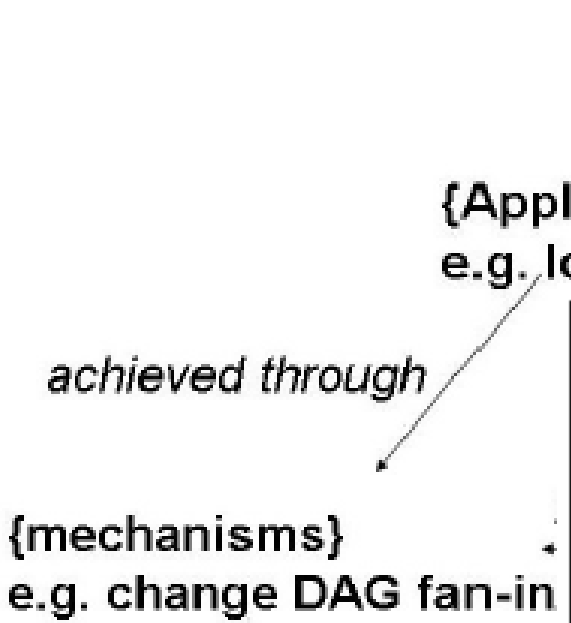
Application Example	Execution Unit	Communication (Data Exchange)	Coordination	Execution Environment
Montage	Multiple sequential and parallel executables	Files	Dataflow (DAG)	Dynamic process creation, workflow execution, file transfer
NEKTAR	Multiple concurrent instances of single executable	Messages	SPMD	MPI, co-scheduling
Coupled Fusion Simulation	Multiple concurrent parallel executables	Stream-based	Dataflow	Co-scheduling, data streaming, async. data I/O
Asynchronous Replica-Exchange	Multiple sequential and/or parallel executables	Pub/sub	Dataflow and events	Decoupled coordination and messaging, dynamic task generation
Climate-Prediction.net (generation)	Multiple sequential executables, distributed data stores	Files and messages	Master/worker, events	AtHome (BOINC)
Climate-Prediction.net (analysis)	A sequential executable, multiple sequential or parallel executables	Files and messages	Dataflow (Forest)	Dynamic process creation, workflow execution
SCOOP	Multiple different parallel executables	Files and messages	Dataflow	Preemptive scheduling, reservations

Types of systems and software tools used …

Application example	Execution unit	Communication	Coordination	Execution environment
Montage	-	-	DAGMan	Pegasus (abstract workflow); Condor DAGMan (for enactment)
NEKTAR	MPICH-G2	MPICH-G2	MPICH-G2	Distributed MPI (e.g., MPICH-G2), cosche-duling (e.g., HARC for advanced reservation)
Coupled fusion simulation	MPI/ OpenMP	PORTALS/ VERBS (over RDMA)	Data I/O: DART/ ADIOS, Coupling: SIENE (over sockets), Kepler	DMA-based data extraction, data streaming tool, puts (gets) on remote memory
Asynch. Replica-exchange mole-cular dynamics	MPI/ pthreads	Meteor (pub/sub notification over JXTA pipes)	Distr. shared space: COMET (over JXTA)	JXTA-based services, COMET (decentralized tuple space implementation) and Meteor
ClimatePrediction.net (generation)	-	BOINC (over http)	BOINC	At-Home (BOINC)
ClimatePrediction.net (analysis)	-	-	Martlet	Workflow execution through Martlet
SCOOP	-	LDM [43]	-	SPRUCE, HARC

AI-supported Tools Interoperability …

- Large number of (potentially unstable) programming systems, tools and deployment environments
  - Understanding how applications make use of these
  - What are the common themes across these?
- Applications are hard to extend
  - Develop them in such a way that it is easy to “swap in/out” coordination capability
  - Analogous to development of operating system services
- Need for abstractions to support application development
  - Formulation, development and execution less dependent on the infrastructure used

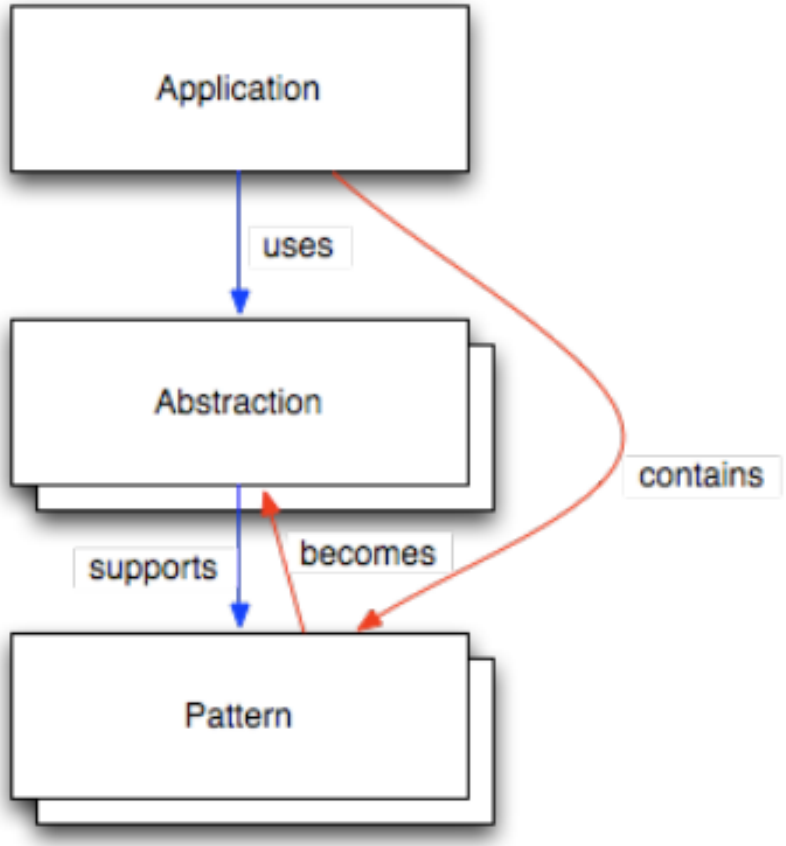


Application Objective	Autonomic Strategy
Load Balancing	**1. Adapt task mapping granularity based on system capabilities/state** File staging, File splitting/merging Task rescheduling, Task migration File distribution and caching, Storage Management **2. Resource Selection** Resource selection (using BQP), resource configuration update, Task rescheduling, Task migration File distribution and caching Storage Management
Scientific Fidelity	**Algorithmic Adaptivity** Change solvers -

- Application-level Objective (AO): Us
  increase throughput, reduce task fai
- Mechanism: action used by applicat
  mechanism m: ($\{m_i\}$, $\{m_i^e\}$, $\{m_o\}$, {
  - $\{m_i\}$ and $\{m_o\}$: file referen
  - $\{m_i^e\}$: input events that trigger start of file staging
  - $\{m_o^e\}$: output events after file staging is completed.
- Strategy: consists of a collection of mechanisms – manual or dynamically constructed by an autonomic approach

What abstractions do we give AI?

- Relation between Application, Abstractions and Patterns:
  - Application: Need or can use >1 R
  - Patterns: Formalizations of commonly occurring modes of computation, composition, and/or resource usage
    - Devel, Deploy & Exec Phase
  - Abstractions: Process, mechanism or infrastructure to support a commonly occurring usage



Coordination	Deployment
Master-Worker (TF, BoT)	Replication
All-Pairs	Co-allocation
Data Processing Pipeline	Consensus
MapReduce	Brokering
AtHome	
Pub-Sub	
Stream	

Concluding remarks …

- Programmable infrastructure opens greater potential for use of AI-based approaches
  - What parameters are tuned by the AI environment
  - Outcome of these on the resulting application
- Supporting AI in hardware –means to accelerate the performance of AI algorithms (particularly neural networks)
- Writing applications with AI in mind
  - Use of abstractions and design patterns that are “enablers” for AI-based techniques