IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Adaptive Workload Management in Db2 Warehouse

David Kalmuk

IBM

Session code: D10

Wednesday November 7th, 9:40am

Db2 for Linux, Unix, Windows

Please Note:

- IBM’s statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM’s sole discretion.
- Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.
- The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.
- The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.
- Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user’s job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Objectives

- Learn about the new Adaptive Workload Management technology and how it automatically manages scheduling and execution of your workload to ensure stability and maximize performance.
- Learn how you can easily assign system resource targets to different workloads to ensure they can meet their performance goals.
- Learn how to monitor your workload performance, activity, and resource consumption to ensure they are meeting their objectives.

Agenda

- Workload Management Basics and the Db2 Workload Manager
- The Challenge of Modern Analytic Workloads
- Db2’s New Adaptive Workload Management Technology
- The Adaptive WLM User Model

Workload Management Goals for a Database System

- Ensure System Stability and Responsiveness
  - Don’t overcommit the system but ensure it’s well utilized
  - Schedule jobs appropriately to ensure fairness and appropriate responsiveness
- Workload Prioritization / Isolation
  - Allow resources to be subdivided between workloads for prioritization / isolation purposes
- Workload Governance and Monitoring
  - Allow definition of rules to govern workloads / detect and abort rogue jobs
  - Allow workload level monitoring

Db2 Workload Manager

- A mature and highly customizable set of capabilities for workload management
  - *Classification, mapping, concurrency control, governance thresholds, resource control*
- View it as a framework with a comprehensive set of ‘tools’ for DIY workload management
  - *Construct nearly any workload management setup you can imagine*
- WLM Best Practices provide a template for building a recommended configurations for managing a warehouse environment
  - *Further refinements add scenarios for isolation, prioritization, production shifts*

The Db2 Workload Manager Menu

Domain	Options
Workload Classification	WORKLOAD
Workload Prioritization	SERVICE CLASS
Job Classification	WORK CLASS / WORK ACTION SET Remapping THRESHOLD
Job Prioritization	SERVICE SUBCLASS
Admission + Resource Control	Concurrency THRESHOLD CPU LIMIT + SHARE PREFETCH + BUFFERPOOL PRIORITY
Governance	Predictive + Reactive THRESHOLD
Monitoring	SQL Functions (Workload, Service class) Event Monitors (Statistics, Activity)

1 SELECT YOUR PROTEIN

HERO BURGER 100% SEASONED ANGUS BEEF BERETTA	4oz 6oz 8oz	320Cals 480Cals 640Cals	$5.99 $7.59 $9.19
GRILLED CHICKEN BREAST		130Cals	$7.99
CRISPY CHICKEN BREAST		250Cals	$7.99
TURKEY BURGER		220Cals	$7.19
WILD ALASKAN SALMON FILLET		100Cals	$7.99
SOUL BURGER		160Cals	$5.99
ALL BEEF HOT DOG		150Cals	$5.29
COMBO MEAL		ADD 480-810 Cals	ADD $3.99
JUNIOR BURGER WITH BUN		220Cals	$3.29
JUNIOR SHAKE		330-490Cals	$3.49
JUNIOR COMBO		ADD 320-490Cals	ADD $2.80

2 SELECT YOUR BREAD

Sesame Poppy Seed Bun	160 Cals	$0.00
Whole-wheat Flat Bread	170 Cals	$0.00
Ciabatta Bun	240 Cals	$0.59
Multigrain Bun	280 Cals	$0.59
Gluten-free Bun	240 Cals	$1.19

Adults and youth (ages 13 and older) need an average of 2,000 calories a day, and children (ages 4 to 12) need an average of 1,500 calories a day. However, individual needs vary.

3 SELECT YOUR CHEESE & TOPPINGS

CHEESE		
REAL CANADIAN CHEDDAR	90 Cals	$0.99
SWISS CHEESE	110 Cals	$1.29
BLUE CHEESE	100 Cals	$1.29
SMOKED CHEDDAR	130 Cals	$1.29
GOAT CHEESE	100 Cals	$1.29

TOPPINGS		
CRISPY ONIONS	150Cals	$0.99
SAUTÉED ONIONS	30 Cals	$0.99
FIRE ROASTED PEPPERS	5 Cals	$0.99
FRIED EGG	80 Cals	$0.99
GUACAMOLE	40 Cals	$0.99
STRIP BACON	60 Cals	$1.29
PORTOBELLO MUSHROOMS	50 Cals	$1.29
BEEF BACON	90 Cals	$1.59

CONDIMENTS		SAUCES	
Ketchup	20Cals	Ancho Chipotle	50Cals
Mustard	3Cals	Cranberry	40Cals
Relish	20Cals	Hero Certified Sauce	70Cals
Slawsa	20Cals	Hero Hot Sauce	2Cals
Lettuce	0Cals	Honey Dijon	20Cals
Tomato	4Cals	Low Fat Mayonnaise	40Cals
Red Onion	4Cals	Maple Chipotle BBQ	30Cals
Jalapeno	20Cals	Mango	30Cals
Sliced Pickle	2Cals		

CONDIMENT OR SAUCE FOR DIPPING 0-70Cals $0.79

4 SELECT YOUR SIDES & EXTRAS

ULTIMATE FRIES	480Cals	$3.29
SWEET POTATO FRIES	520Cals	$3.99
CRUNCHY CHICKEN STRIPS	350Cals	$5.99
ONION RINGS	410Cals	$3.99
POUTINE	800Cals	$5.99
TEMPURA ZUCCHINI	350Cals	$3.99
GRAVY	30Cals	$0.79
GREEN SALAD	15Cals	$5.99
ADD CHICKEN 130 Cals SALMON 100 Cals		$3.50

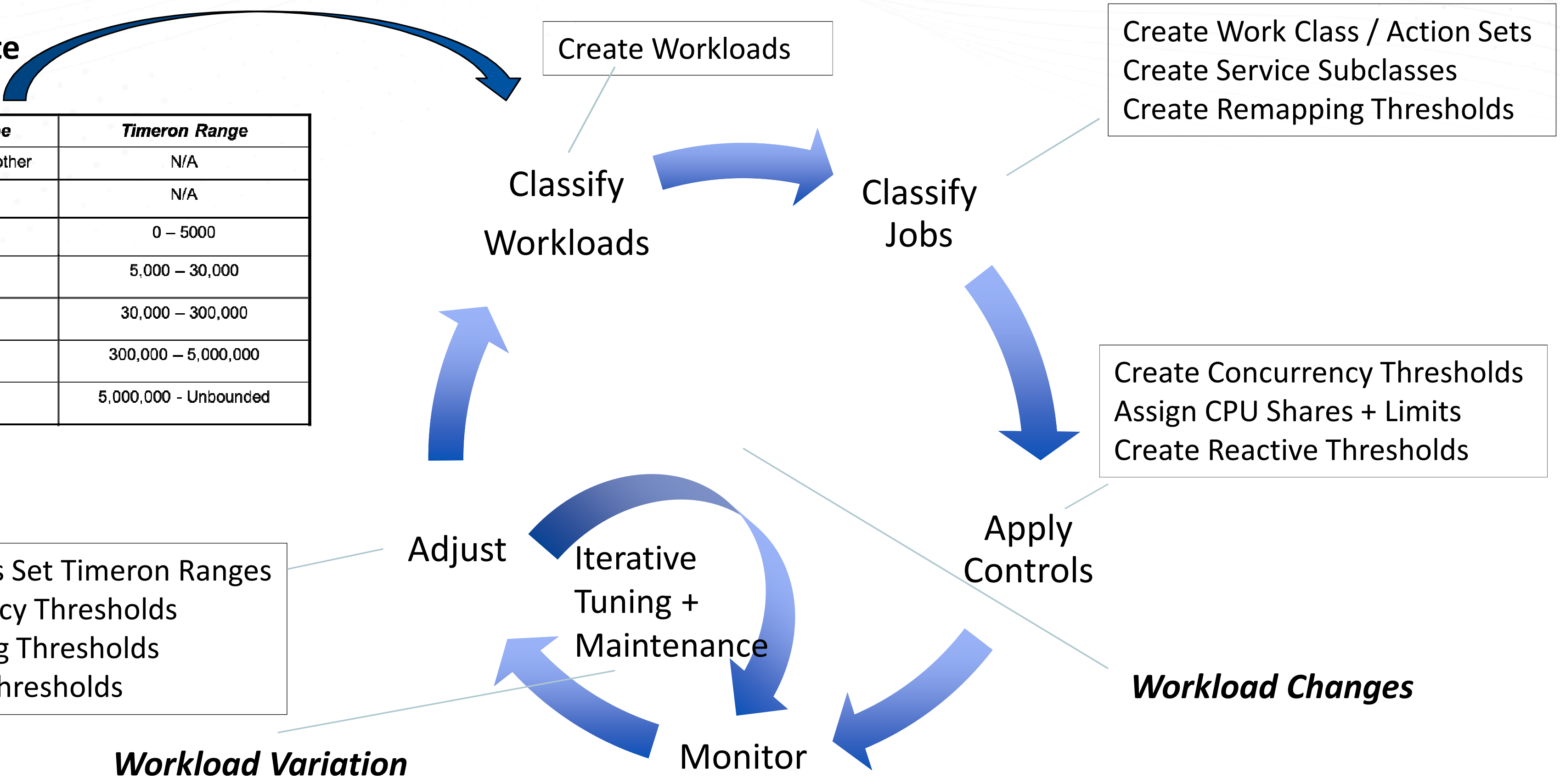
REAL SHAKES & DESSERTS		
VANILLA SHAKE	590Cals	$5.49
CHOCOLATE SHAKE	770Cals	$5.49
STRAWBERRY SHAKE	750Cals	$5.49
ZIPP Smarties	670Cals	$3.99
ZIPP Oreo	560Cals	$3.99
ICE CREAM CONE	330Cals	$2.29
CARAMEL SUNDAE	600Cals	$3.99
CHOCOLATE SUNDAE	500Cals	$3.99
STRAWBERRY SUNDAE	480Cals	$3.99

BEVERAGES		
20oz bottomless	1-330Cals	$2.69
Bottled Water	0Cals	$2.49
Bottled Drinks	0-230Cals	$2.89

Db2 WLM Best Practices Configuration Lifecycle

Best Practice Template

Service Subclass	*Work Type*	*Timeron Range*
Default	CALL, DDL, other	N/A
LOAD	LOAD	N/A
Trivial	DML	0 – 5000
Minor	DML	5,000 – 30,000
Simple	DML	30,000 – 300,000
Medium	DML	300,000 – 5,000,000
Complex	DML	5,000,000 - Unbounded



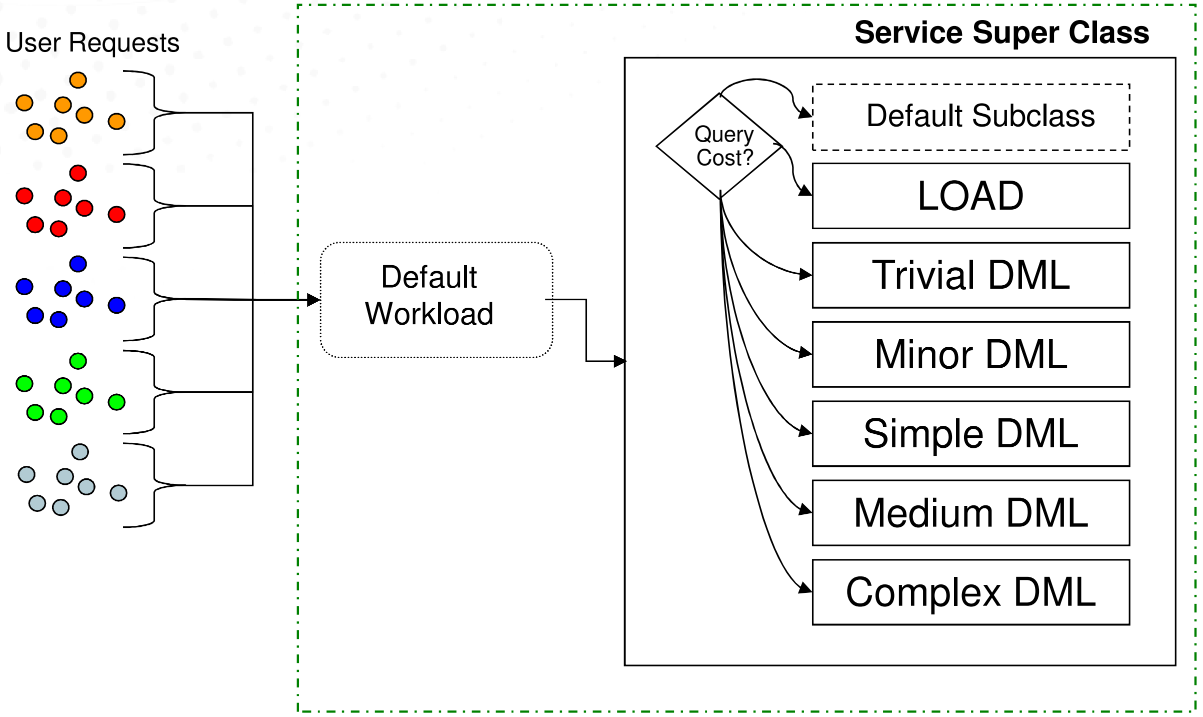
Create Workloads

Create Work Class / Action Sets
Create Service Subclasses
Create Remapping Thresholds

Create Concurrency Thresholds
Assign CPU Shares + Limits
Create Reactive Thresholds

Adjust Work Class Set Timeron Ranges
Adjust Concurrency Thresholds
Adjust Remapping Thresholds
Adjust Reactive Thresholds

System Stability BP Configuration



User Requests

Default Workload

Service Super Class

Query Cost?

Default Subclass

LOAD

Trivial DML

Minor DML

Simple DML

Medium DML

Complex DML

Work action set timeron costs subdivide work into “lanes” based on cost for tiered job scheduling to provide consistent throughput

Subclass concurrency limits control mix of work and overall admission

Balance total work, division between subclasses

a < Timeron Cost <= b

Concurrency Limit = x

b < Timeron Cost <= c

Concurrency Limit = y

c < Timeron Cost <= d

Concurrency Limit = z

d < Timeron Cost <= e

Concurrency Limit = q

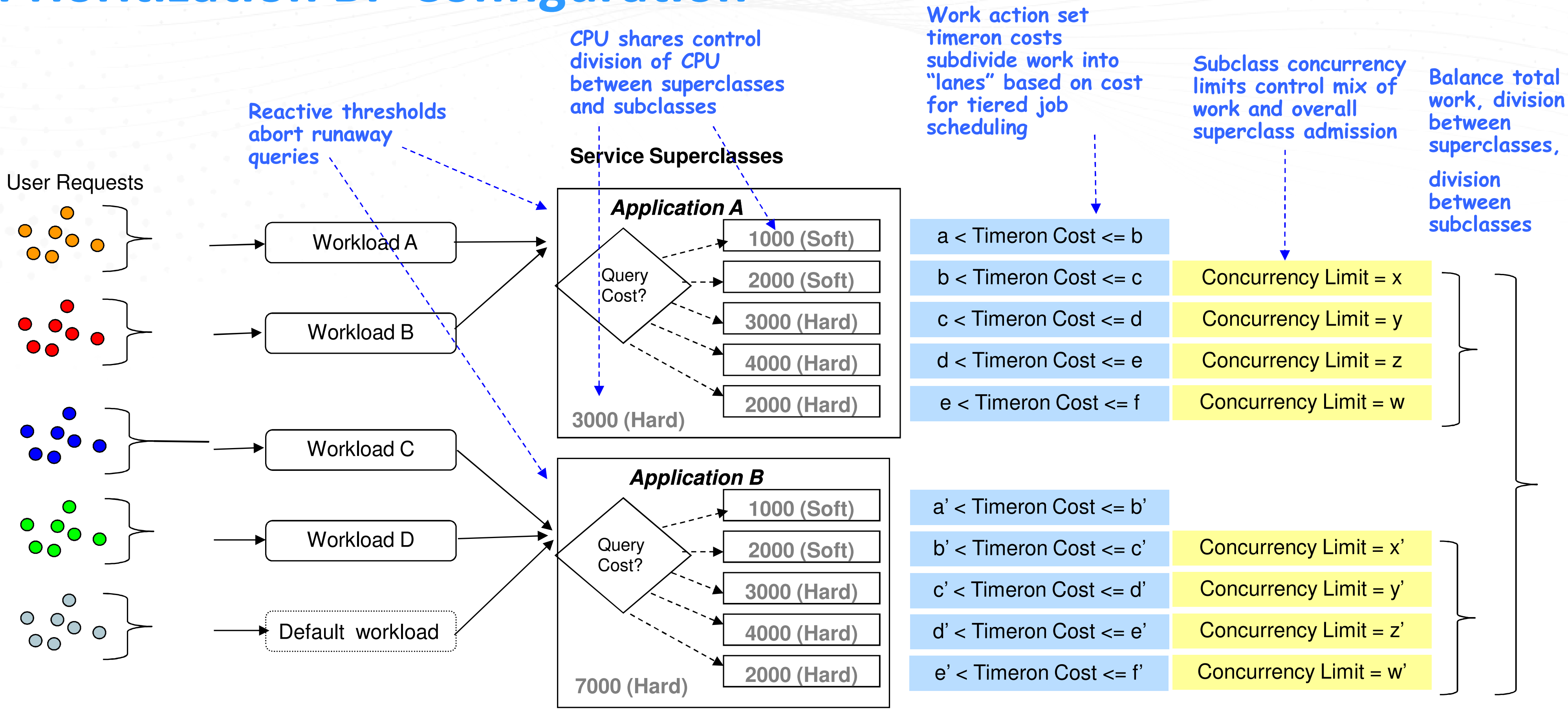
e < Timeron Cost <= e

Concurrency Limit = r

f < Timeron Cost <= f

Concurrency Limit = s

Prioritization BP Configuration



Reactive thresholds abort runaway queries

CPU shares control division of CPU between superclasses and subclasses

Work action set timeron costs subdivide work into “lanes” based on cost for tiered job scheduling

Subclass concurrency limits control mix of work and overall superclass admission

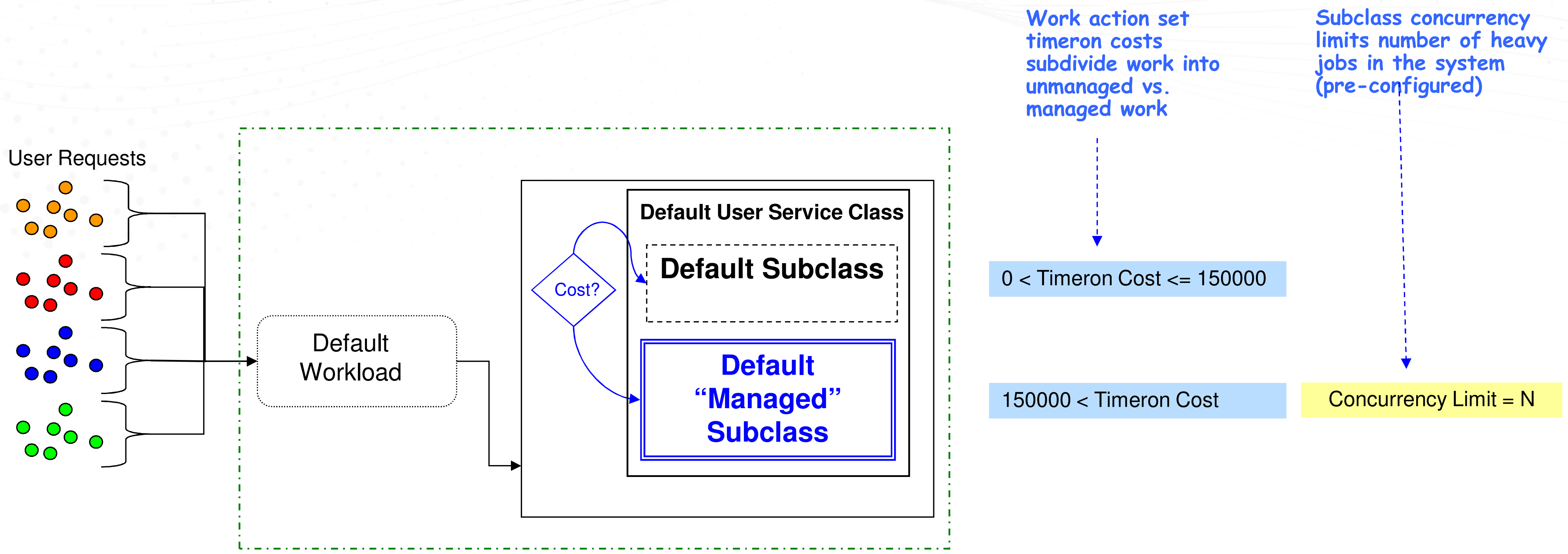
Balance total work, division between superclasses,

division between subclasses

a < Timeron Cost <= b	
b < Timeron Cost <= c	Concurrency Limit = x
c < Timeron Cost <= d	Concurrency Limit = y
d < Timeron Cost <= e	Concurrency Limit = z
e < Timeron Cost <= f	Concurrency Limit = w

a’ < Timeron Cost <= b’	
b’ < Timeron Cost <= c’	Concurrency Limit = x’
c’ < Timeron Cost <= d’	Concurrency Limit = y’
d’ < Timeron Cost <= e’	Concurrency Limit = z’
e’ < Timeron Cost <= f’	Concurrency Limit = w’

Default BLU ANALYTICS Stability Configuration



Work action set timeron costs subdivide work into unmanaged vs. managed work

Subclass concurrency limits number of heavy jobs in the system (pre-configured)

0 < Timeron Cost <= 150000

150000 < Timeron Cost

Concurrency Limit = N

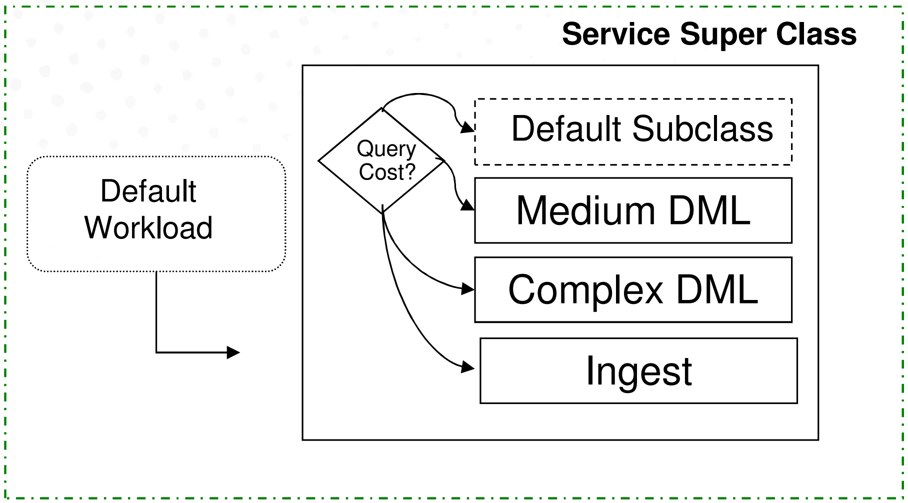
Query Costs and Concurrency Limits

- Maintaining this type of WLM configuration involves manual processes that can be fairly labor intensive
- The underlying reason is that both **query cost ranges** and **concurrency limits** are lower level and **indirect** controls over what we are actually trying to manage
  - Query cost = Use estimate of query complexity to differentiate based on **response time**
  - Concurrency limit = Control **resource consumption** for jobs in a particular class via fixed limit
- Most database vendors use similar techniques with similar complexities - why?
  - Eg. “Concurrency thresholds”, “Throttles”, “Slots”, ”Queues”, “Memory limits”, etc.
- Predicting **response times** and **resource consumption** accurately enough to be actionable is **hard**!
- **Fixed limits** are **much easier** to implement from a technology perspective.

The Challenge of Modern Analytic Workloads

- Diverse range of jobs from miniscule point lookups to massive analytic queries
- Highly dynamic workloads combining
  - High volumes of operational point queries
  - Concurrent complex analytic queries of varying shapes and sizes
  - Continuous data ingest
- With in-memory column store technologies fixed resources like memory become the limiting factor vs. CPU
  - Much less forgiving if system gets overcommitted
- For these types of workloads configurations based on fixed limits are necessarily sub-optimal and difficult to tune

Trying to tune a mixed workload configuration…



0 < Timeron Cost <= ??	Concurrency Limit = ??
?? < Timeron Cost <= ??	Concurrency Limit = ??
?? < Timeron Cost <= ??	Concurrency Limit = ??
	Concurrency Limit = ??

For response time < 30 seconds target 20% resources

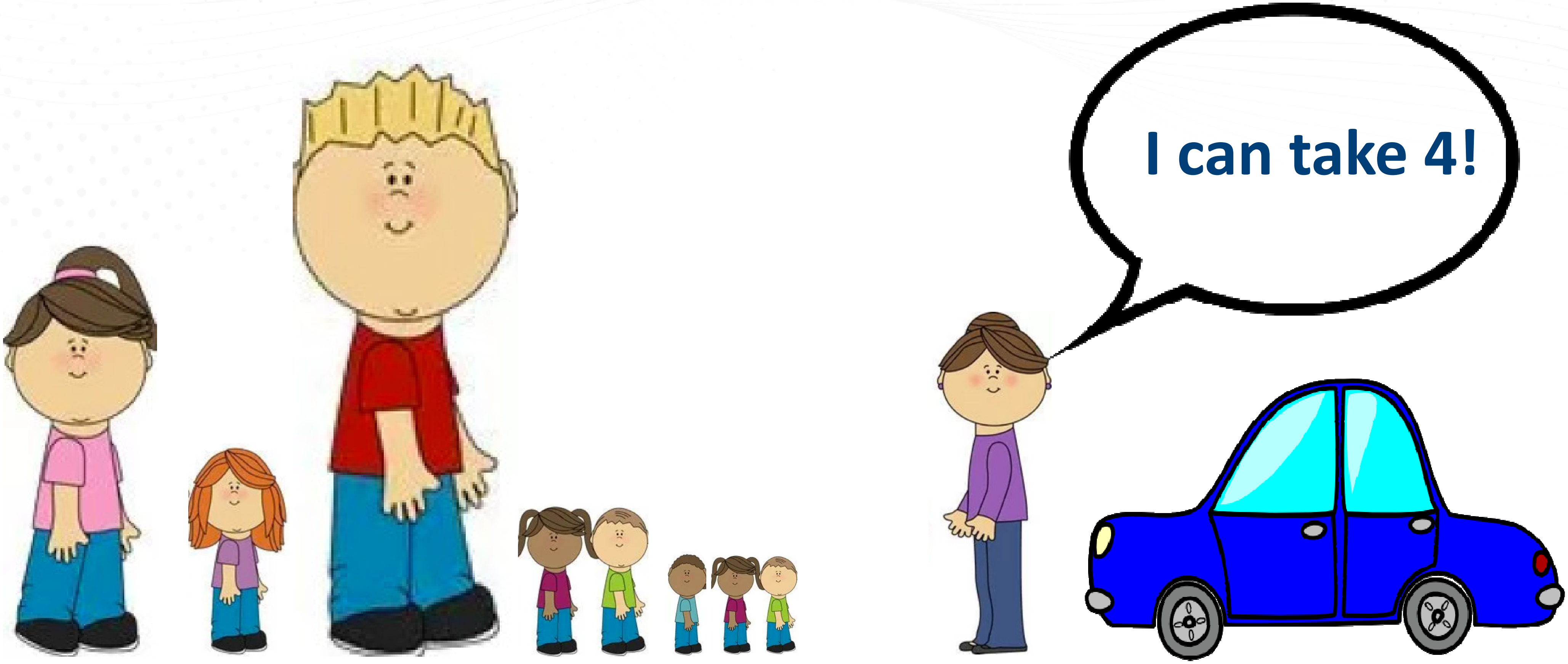
For response time < 600 seconds target 20% resources

For response time > 600 seconds target 30% resources

For ingest target 30% of resources

Indirect controls; onus is on the user to derive, apply, and adjust to maintain appropriate fixed limits.

The problem…



Db2’s New Adaptive Workload Management

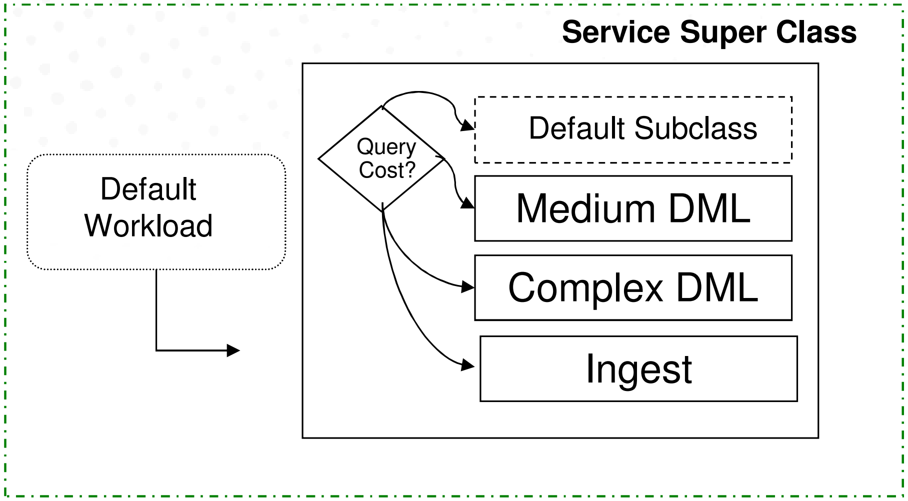
Db2's New Adaptive Workload Management Technology

- Admission based on query resource footprint and fit rather than fixed concurrency limits
  - Adjusts admission implicitly based on the workload without manual tuning
  - More intelligent job scheduling makes more efficient use of system resources
  - Improved performance for concurrent workloads
- Resources to be considered by adaptive admission control
  - Sort memory (aka query working memory)
    - Key resource bottleneck for BLU column store
  - CPU load impact / number of threads
    - Control admission to target a healthy CPU load based on expected query degree
- Initially available in Db2 Warehouse on Cloud, Db2 Warehouse, IIAS
  - Db2 software support will follow

Adaptive Workload Management Benefits

- Deliver true automatic workload management out of the box with zero tuning
- Removes need to configure + tune fixed concurrency limits
- Improved stability and performance
- Enables much simpler and more powerful admission models

Mixed Workload Configuration under Adaptive WLM



For response time < 30 seconds target 20% resources

0 < Time Estimate <= 30s	Resource Share 20%
30s < Time Estimate <= 600s	Resource Share 20%
600s < Time Estimate <= INF	Resource Share 30%
	Resource Share 30%

For response time < 300 seconds target 20% resources

For response time > 300 seconds target 30% resources

For ingest target 30% of resources

Direct targets; onus is on the system to manage system and constantly adjust behavior to meet targets (innovation required)

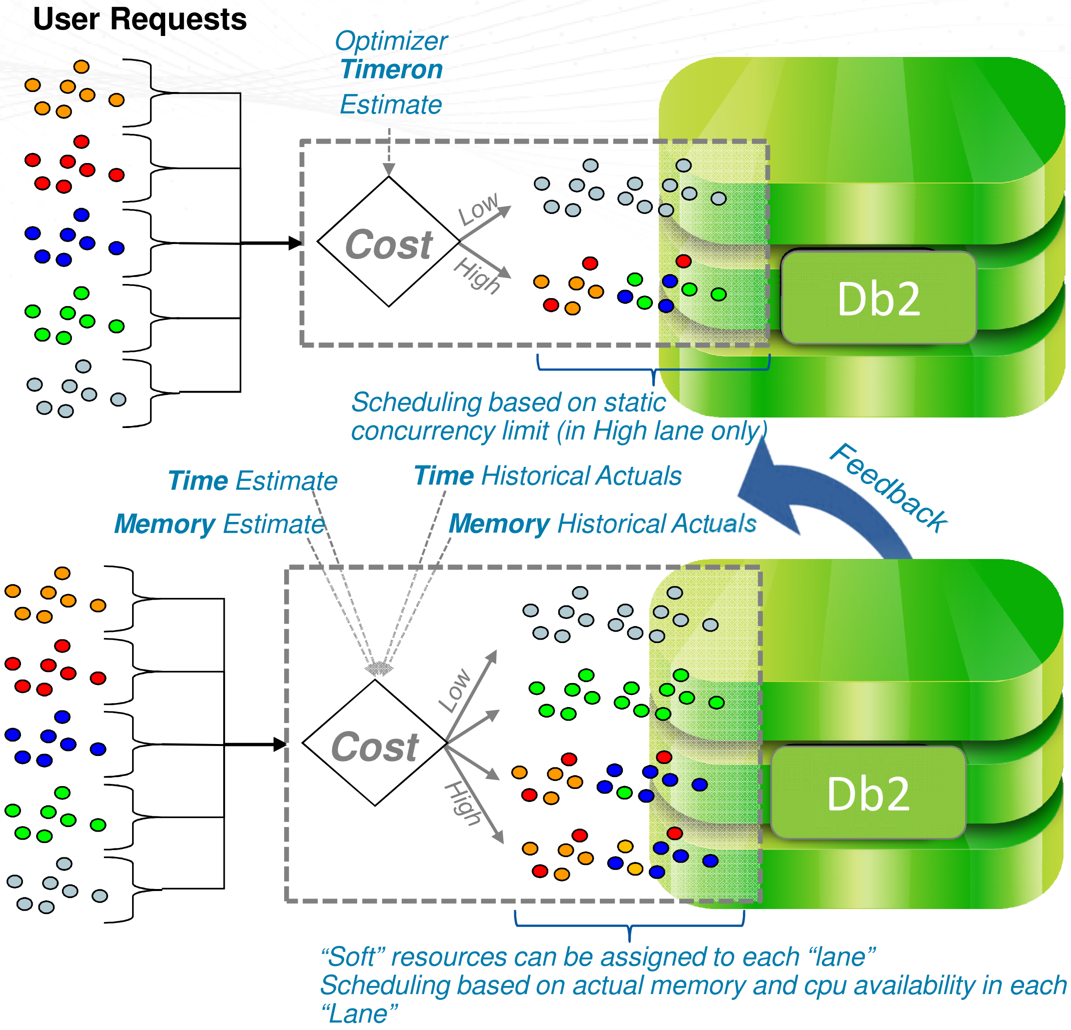
Intelligent Job Scheduling

Current Automatic WLM

- Cost evaluation includes only “timeron” estimate
- Open ended (no feedback)
- Scheduling based on static concurrency threshold

New Adaptive WLM

- Cost evaluation includes memory & cpu load & time duration
- Includes historical feedback based on past executions
- Scheduling based on dynamic view of resource availability in each “lane”
- Expected benefits
  - ***- Improved robustness under high load***
  - ***- Improved SLA achievement***
  - ***- Improved overall resource efficiency & throughput***

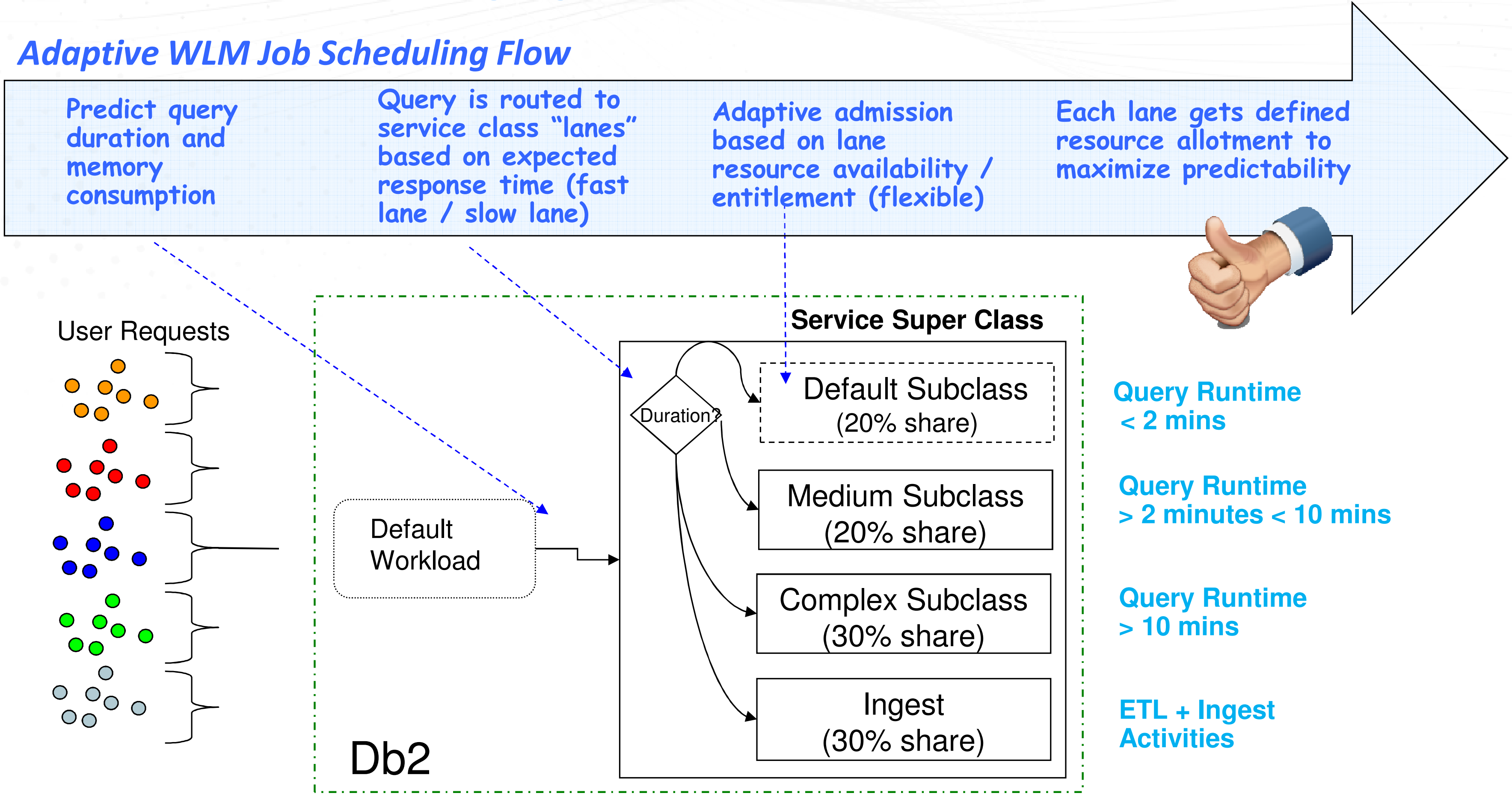


Scheduling based on static concurrency limit (in High lane only)

“Soft” resources can be assigned to each “lane”
Scheduling based on actual memory and cpu availability in each “Lane”

Under the Hood: Managing Mixed Workloads for Predictable Performance

Adaptive WLM Job Scheduling Flow



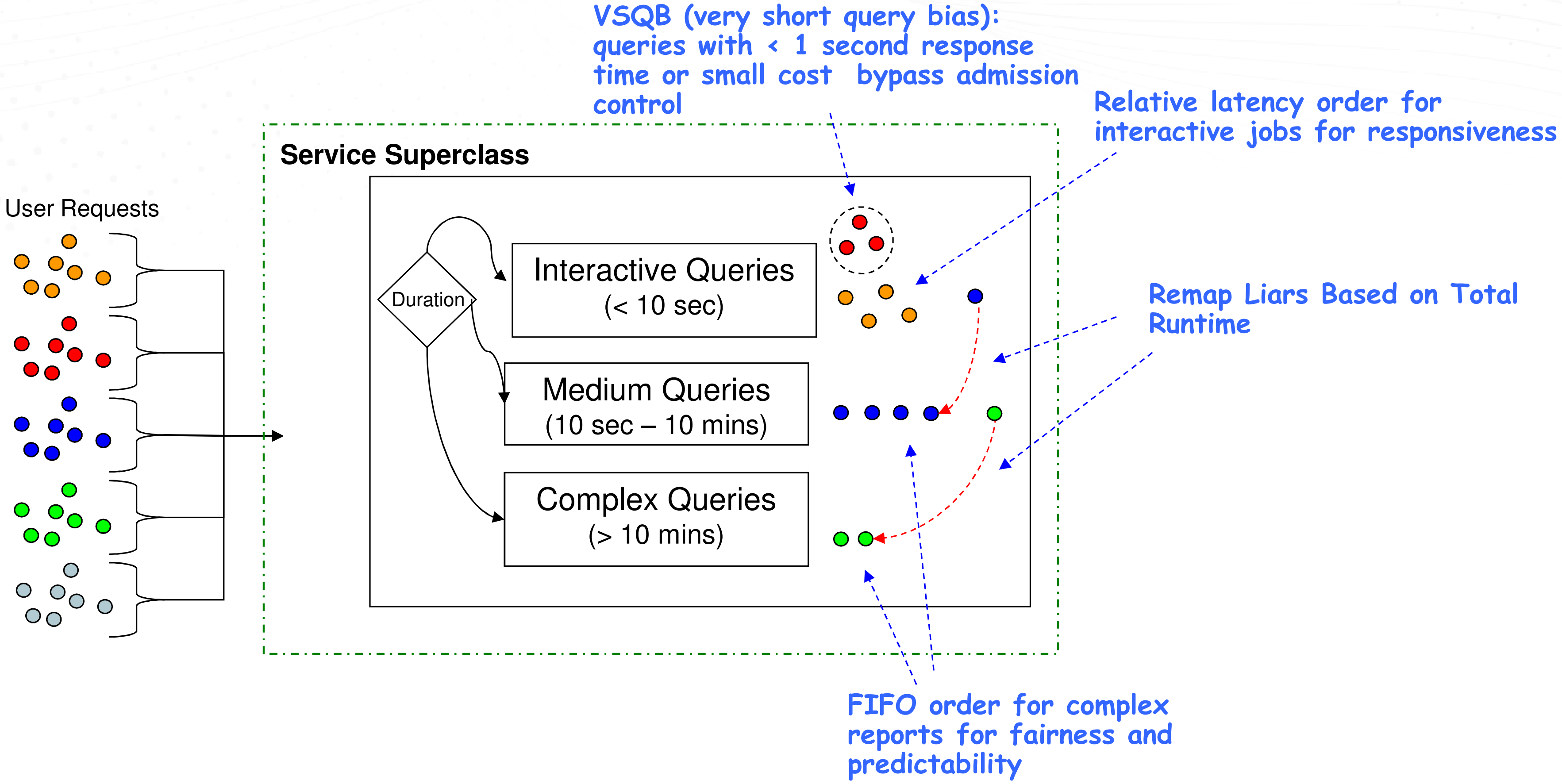
Predict query duration and memory consumption

Query is routed to service class "lanes" based on expected response time (fast lane / slow lane)

Adaptive admission based on lane resource availability / entitlement (flexible)

Each lane gets defined resource allotment to maximize predictability

Under the Hood: Latency Oriented Job Scheduling



VSQB (very short query bias): queries with < 1 second response time or small cost bypass admission control

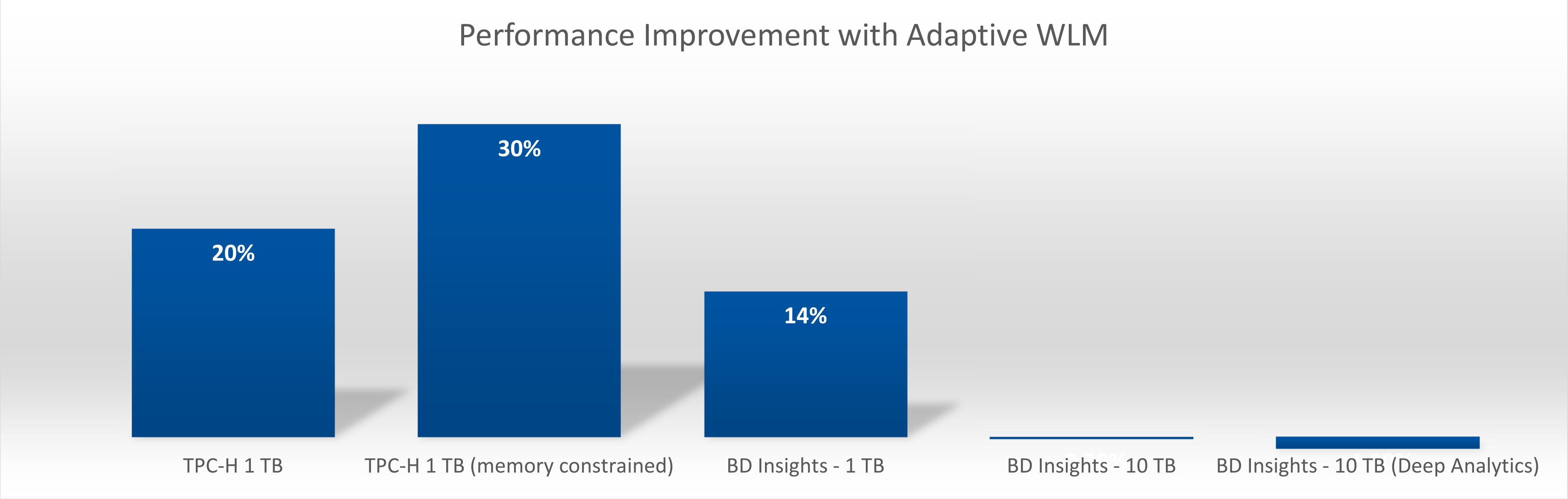
Relative latency order for interactive jobs for responsiveness

Remap Liars Based on Total Runtime

FIFO order for complex reports for fairness and predictability

Some Performance Numbers

- Analytical workloads performed at par or better with Adaptive WLM compared to current WLM using **default** concurrency thresholds across a set of internal workloads
  - More optimal amount of work is admitted into the system based on CPU Load and memory consumption. Less thrashing leads to better performance!
  - System stability maintained by avoiding overcommitting the system



Monitoring Adaptive WLM

- Current working memory usage per partition

```
with sortmem (sheapthresshr, sheapmember) as
(select value, member from sysibmadm.dbcfg where name = 'sheapthres_shr')
select member, sort_shrheap_allocated as allocated_mem, sheapthresshr as configured_mem
from table(mon_get_database(-2)) as t, sortmem
where sheapmember = member;
```

- Average statement execution time and resource usage

```
with sortmem (sheapthresshr, member) as
(select value, member from sysibmadm.dbcfg where name = 'sheapthres_shr')
select p.member, wlm_queue_time_total, coord_stmt_exec_time, num_executions,
       adm_bypass_act_total, query_cost_estimate, estimated_runtime,
       estimated_sort_shrheap_top * 100 / sheapthresshr as estimated_sort_pct,
       sort_shrheap_top * 100 / sheapthresshr as sort_pct,
       substr(stmt_text,1,256) as stmt
from table(mon_get_pkg_cache_stmt(null,null,null,-2)) p,
sortmem s where p.member=s.member;
```

Monitoring Adaptive WLM

- Currently executing and queued statements with details

```
with sortmem (sheapthresshr, member) as
(select value, member from sysibmadm.dbcfg where name = 'sheapthres_shr')
select b.application_name, b.session_auth_id, a.entry_time, a.local_start_time,
       a.activity_state, a.query_cost_estimate, a.estimated_runtime,
       a.effective_query_degree, a.adm_bypassed,
      (a.estimated_sort_shrheap_top * 100) / c.sheapthresshr as mem_estimate_pct,
      (a.sort_shrheap_top * 100) / c.sheapthresshr as peak_mem_used_pct,
       substr(a.stmt_text, 1, 512) as stmt_text
from table(mon_get_activity(null,-2)) as a,
     table(mon_get_connection(null,-1)) as b,
     sortmem as c
where (a.application_handle = b.application_handle)
order by activity_state;
```

Monitoring Adaptive WLM

- (cont'd)

Memory estimates used for admission

Very short query admission bypass

Peak memory usage

```
… ACTIVITY_STATE QUERY_COST_ESTIMATE ESTIMATED_RUNTIME EFFECTIVE_QUERY_DEGREE ADM_BYPASSED MEM_ESTIMATE_PCT PEAK_MEM_USED_PCT …
  -------------- ------------------- ----------------- ---------------------- ------------ ---------------- -----------------
  EXECUTING      58                  36733             24                      1            5.14355          4.95233
  EXECUTING      58342               267330            24                      0            3.14355          4.12342
  EXECUTING      58423442            136733            24                      0            11.14355         8.95233
  EXECUTING      182235523           5367333           24                      0            7.14355          9.95233
  QUEUED         679342340083        104336733         24                      0            75.14355         0.00
```

Queued job waiting for admission

Adaptive WLM Configuration

- Out-of-the-box configuration is designed to be largely autonomous + adaptive with no tuning requirements
- One optional tunable that you should be aware of is the `WLM_AGENT_TRGT_LOAD` database configuration parameter
- This parameter controls the maximum thread load per core that the workload manager will allow into the system at a time to avoid degrading processing efficiency.
- The thread load per core on the database is computed as the sum of the `DEGREE` of all the queries executing on the system.
- Example:
  - Running 6 queries with DEGREE=12 on a 12-core system results in a thread load per core of 6
  - Running 24 queries with DEGREE=1 on a 12-core system results in a thread load per core of 2

Adjusting WLM_AGENT_TRGT_LOAD

- The default `WLM_AGENT_TRGT_LOAD` is computed based on the system hardware and should be optimal for most scenarios
- Consider increasing the `WLM_AGENT_TRGT_LOAD` if:
  - The workload manager is queueing jobs AND
  - There is sufficient sort memory to accommodate more jobs AND
  - None of the system resources are saturated (CPU, I/O, network)
- Consider decreasing the `WLM_AGENT_TRGT_LOAD` if:
  - The system is running a concurrent workload AND
  - The CPU run queues on the system are very heavily loaded and it's degrading system throughput
- Example:

```
UPDATE DB CFG FOR MYDB USING WLM_AGENT_TRGT_LOAD 24
```

Adjusting SORTHEAP and SHEAPTHRES_SHR

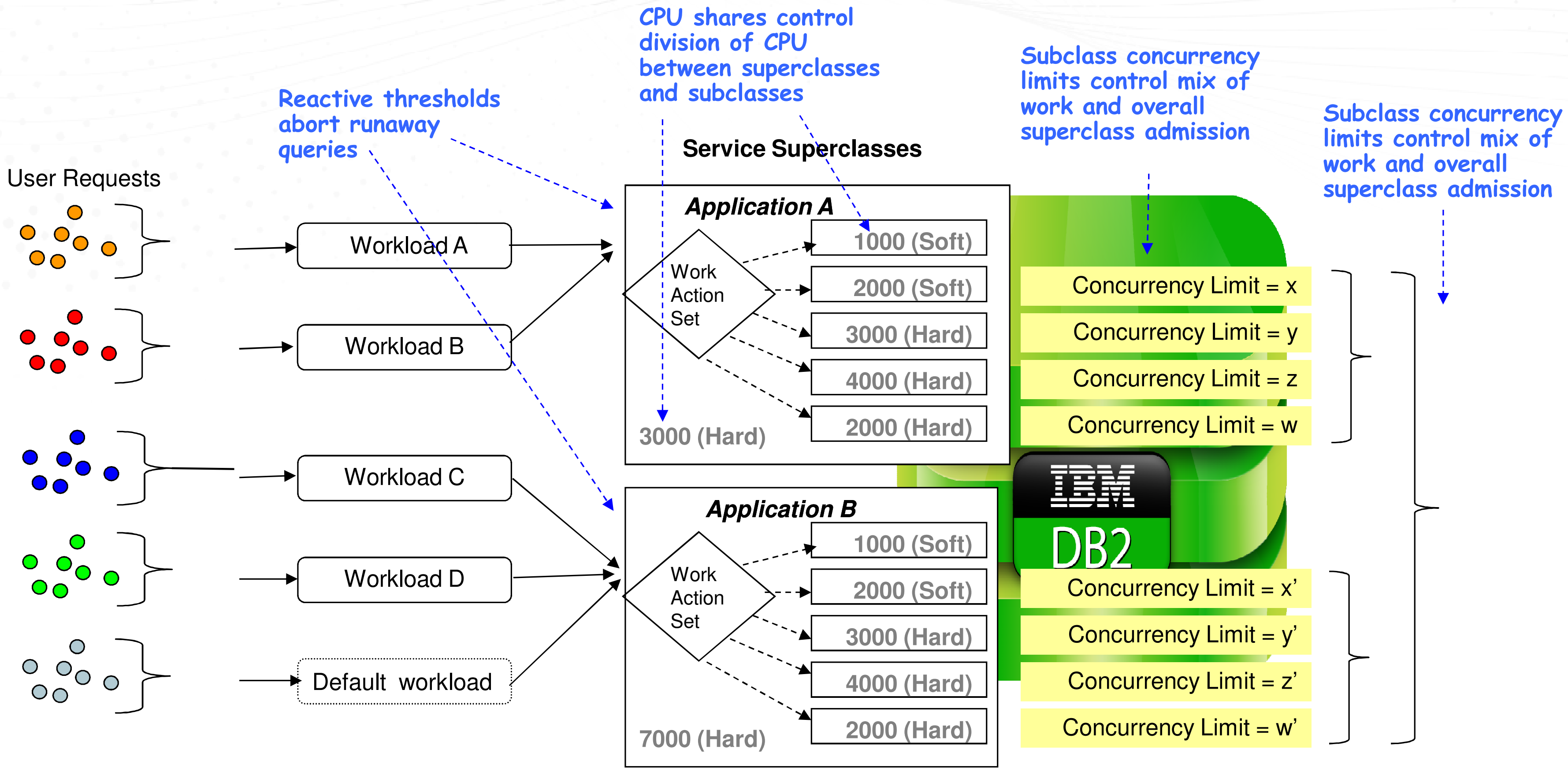
- Since Adaptive WLM manages admission based on query resource demands altering the working memory configuration will have a direct impact on job scheduling behavior
- Increasing `SORTHEAP` relative to `SHEAPTHRES_SHR`
  - Allows more memory per operator (and by extension query) reducing execution time, but fewer jobs will be able to run simultaneously
- Decreasing `SORTHEAP` relative to `SHEAPTHRES_SHR`
  - Allows less memory per operator (and by extension query) increasing execution time, but more jobs will be able to run simultaneously
- Increasing `SHEAPTHRES_SHR` by trading off `BUFFERPOOL` memory
  - This strategy can allow increased concurrency without otherwise sacrificing individual query performance
  - Useful in cases where significant large queries result in concurrency bottlenecks

Adaptive WLM Simplified User Model

Adaptive WLM Simplified User Model

- In addition to its intelligent and autonomous out of the box workload management the Adaptive WLM technology can also enable a much simpler and more powerful user model
- Recall that a lot of the complexity in configuring the Db2 Workload Manager today is the requirement for the user to set and adjust lower level fixed limits to achieve the desired behavior
- By enabling more goal oriented configurations that the system adapts to meet based on the workload we can step up a level of abstraction and create far more user friendly workload management capabilities
- The following section describes our thinking around how we will allow the user to define a customized workload management configuration with Adaptive WLM
  - *Note this is not a commitment to deliver the specified function*

Recap of where we are today….



Reactive thresholds abort runaway queries

CPU shares control division of CPU between superclasses and subclasses

Subclass concurrency limits control mix of work and overall superclass admission

Subclass concurrency limits control mix of work and overall superclass admission

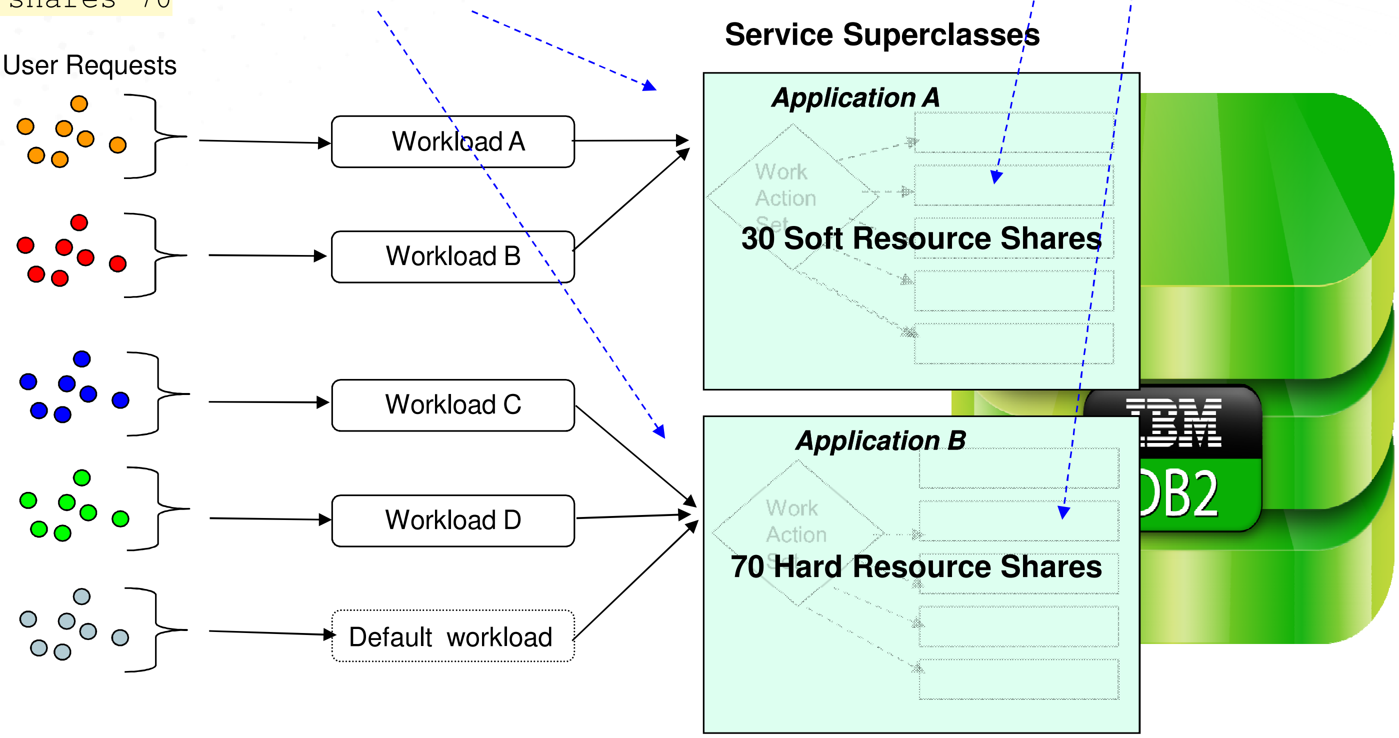
Adaptive WLM User Model (future)

```
create service class sc1
  soft resource shares 30

create service class sc2
  hard resource shares 70
```

Simplified reactive thresholds abort runaway queries

Resource share controls admission and division of runtime resources



Intelligent resource based job scheduling and runtime CPU control ensures fairness + responsiveness and automatically adapts to any workload

More User Model Details (future)

- Create a service superclass pre-configured for one of three defined workload types
  - `INTERACTIVE` for response sensitive jobs
  - `BATCH` for longer running jobs
  - `MIXED` for workloads that run a combination of both
- Assign a resource share to the service class
  - Specifies the proportion of database resources this service class is entitled to
  - Shares can be either `HARD` or `SOFT` for more flexible vs strict resource assignment
- The system does the rest!

An example

- Divide the database resources between 3 distinct workloads
  - High priority interactive reports that require a fast response
  - ETL jobs that need sufficient resources to complete within a specific window
  - Other general purpose tasks on the system that don’t fall into the above categories

```
create service class HIPRI soft resource shares 25 for INTERACTIVE
create service class ETL soft resource shares 25 for BATCH
create service class GENERAL soft resource shares 50 for MIXED
```

```
create workload REPORTS session_user(‘EDW_REPORTS’) service class HIPRI
create workload ETLJOBS session_user(‘EDW_ETL_USER’) service class ETL
alter workload SYSDEFAULTUSERWORKLAOD service class GENERAL
```

Simplified Thresholds

- To complement the simplified service class model we plan to introduce simplified syntax around thresholds to support workload governance
- Example:
  - Current CREATE THRESHOLD DDL

```
CREATE THRESHOLD LONGRUNNINGSQL FOR DATABASE ACTIVITIES
ENFORCEMENT DATABASE WHEN ACTIVITYTOTALRUNTIME > 1 HOUR
STOP EXECUTION;
```

  - Simplified CREATE THRESHOLD DDL

```
CREATE THRESHOLD LONGRUNNINGSQL FOR DATABASE
WHEN ACTIVITYTOTALRUNTIME > 1 HOUR STOP EXECUTION;
```

New Monitoring Functionality

- **SQL Functions**
  - `MON_GET_SERVICE_SUPERCLASS_STATS (future)`
    - Higher level statistics group to match more abstract control levels + additional metrics related to Adaptive WLM behavior
  - `MON_GET_ACTIVITY`
    - Additional metrics to understand Adaptive WLM behavior
    - See also prior examples
- **Event Monitors**
  - `STATISTICS (future)`
    - New logical grouping for superclass statistics + metrics
  - `ACTIVITY`
    - Additional metrics to understand Adaptive WLM behavior

Other Nuts and Bolts

- Adaptive WLM simplifies and abstracts some of the lower level workload manager constructs but it is still fully integrated / compatible with them
- Subclasses / work-class sets / work action sets are still the underlying mechanisms for controlling finer grained job scheduling and resource management
- This last section summarizes the lower level constructs that are being introduced by Adaptive WLM for power users that want to know all the gory details

New WLM objects introduced by Adaptive WLM

- Service superclasses + subclasses (future)
  - Resource share attribute for admission + runtime control
  - Superclass definitions that pre-define subclasses + work class / action sets
- Work class / work action sets
  - New mapping based on query RUNTIME
- Thresholds
  - Simplified threshold syntax (optional)
  - New ACTIVITYTOTALRUNTIME threshold
  - New ACTIVITYTOTALRUNTIMEINALLSC remapping threshold

Summing Up

- Innovative new workload management technology in Db2 Warehouse that automatically adapts to your workload
- Leverages intelligent job scheduling for improved stability and performance
- Simplified user model will allow you to quickly and easily divide database resources between different workloads in order to prioritize and meet your performance goals
- Technology improvements will continue to roll out incrementally

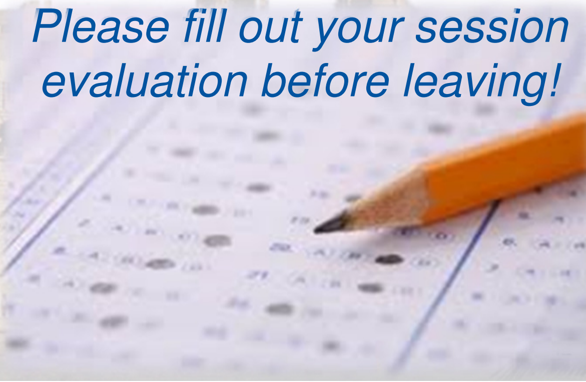
Questions?

David Kalmuk

IBM

dckalmuk@ca.ibm.com

Session code: D10



Please fill out your session evaluation before leaving!