OSIsoft.

vCampus Live! 2013

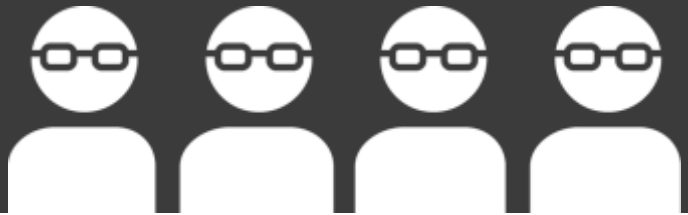
Optimizing PI AF for Performance and Scalability

Presented by **Chris Manhard, AF Engineering Group Leader**
Paul Combellick, AF Server Principal Engineer

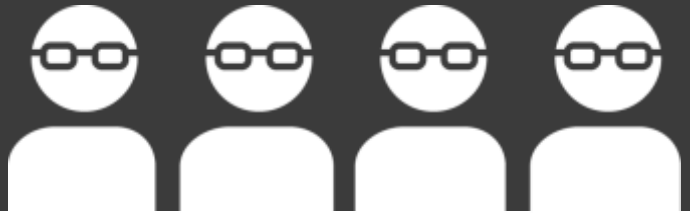
OSIsoft.

vCampus Live! 2013

WHERE PI GEEKS MEET



Optimizing your AF SDK based Application



AF SDK Application Performance

- **The fastest way to PI and AF data**
- **AF SDK does what you ask**
- **It is easy to ask inefficiently**
- **AF SDK does not predict your next call**
- **Large scale performance requires care**

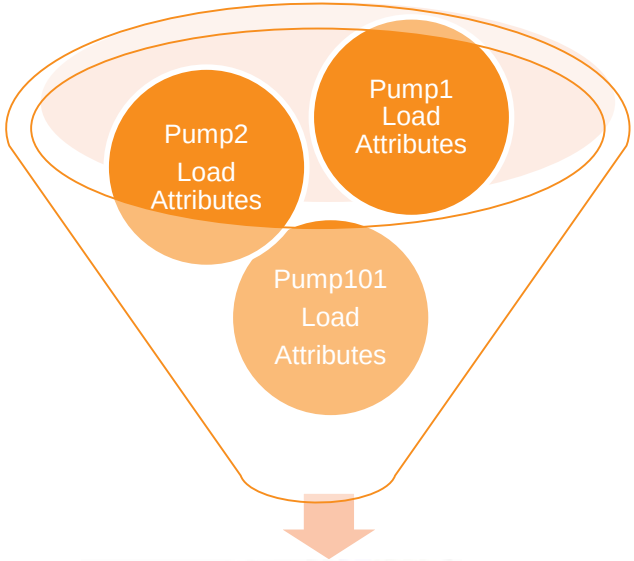
Also, See VCampus 2012 Live! – PI AF SDK Performance and Scalability. Ray Hall

"With Great Power Comes Great Responsibility"
Spiderman's Uncle Ben,

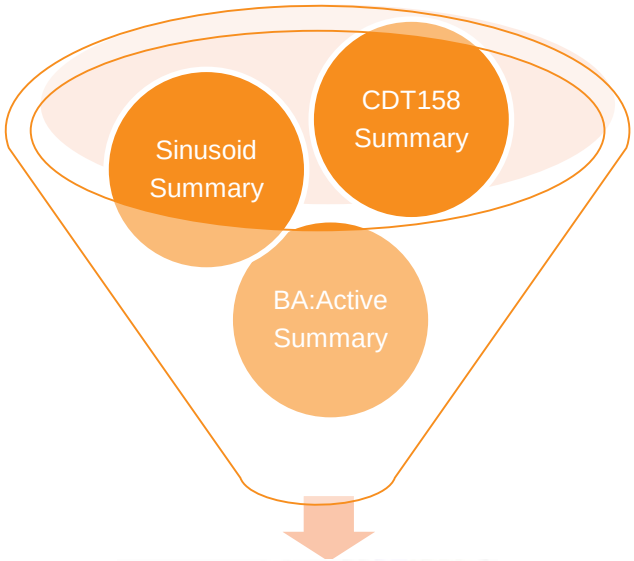
Strategies for best SDK Performance

- Bulk
- Parallel
- Process Change Streams
- Cache on the Client
- Pre-Calculate
- Diagnose Performance Bottlenecks

Make Calls in Bulk

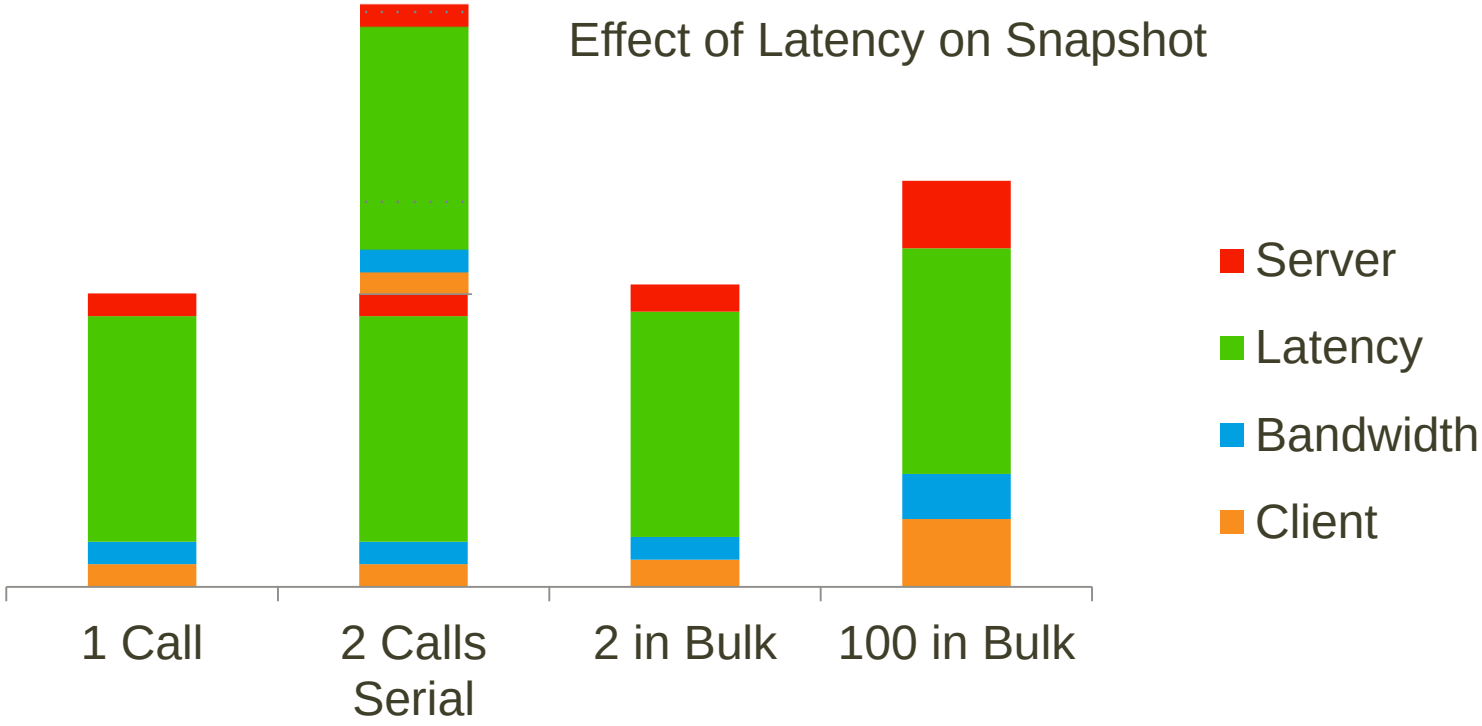


AF Server



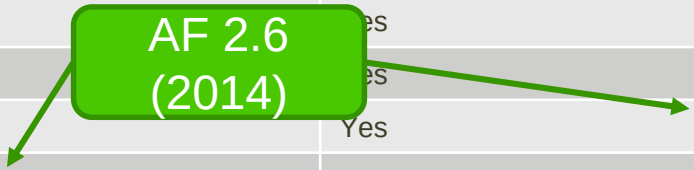
PI Data Archive

Bulk vs. Serial – Performance 101



Getting Data – New Bulk Calls to PI

Capability	PIPoint	PIPointList	AFAttribute / AFData	AFAttributeList / AFListData
Snapshot	Yes	Yes	Yes (AFAttribute.GetValue)	Yes
Recorded Value	Yes	Yes	Yes	Yes
Interpolated Value	Yes	Yes	Yes	Yes
Summary	Yes	**Yes**	Yes	**Yes**
Recorded Values	Yes	**Yes**	Yes	**Yes**
Interpolated Values	Yes	**Yes**	Yes	**Yes**
Plot Values	Yes	**Yes**	Yes	**Yes**
Summaries	Yes	**Yes**	Yes	**Yes**
Filtered Summaries	Yes	**Yes**	Yes	**Yes**
Annotations	Yes	No	Yes (PIPoint DR only)	No
Update Value	Yes	n/a	Yes	n/a
Update Values	Yes	Yes via PIServer	Yes	Yes
Calculated Values	Yes (AFCalculation)	No	Yes (AFCalculation)	No
Data Pipe	NA	Yes (PIDataPipe)	NA	**Yes (AFDataPipe)**



8

New Bulk Call Overview

- New RPC in PI Server 2012
- Allows requests for multiple points to be batched into a single request
- Caller can control how results are paged to the client (by Tag Count, by Event Count)
- Much Faster than serial
- Somewhat faster than making parallel call

New Bulk RPC Example

AFListData.PlotValues

IEnumerable of same return type as non-bulk call

```
C#

public IEnumerable<AFValues> PlotValues(
          AFTimeRange timeRange,
          int intervals,
          PIPagingConfiguration pagingConfig
)
```

Specify how data is paged back from client

Bulk RPC Example - Paging

PIPagingConfiguration parameter

Properties

	Name	Description
	BulkPayloadPercentThreshold	This value represents a percentage of the entire RPC that must be completed to cause a page to be proactively returned to prevent the operation timeout from expiring.
	KeepAliveTimeout	This value is the maximum amount of time allowed to elapse between calls to get the next page of results.
	OperationTimeoutOverride	This value overrides the operation timeout set on the PIServer for the duration of the data access call. The operation timeout is the maximum amount of time that can elapse on the **PIServer** while fetching each page.
	PageSize	The size of the pages that will be returned depending on the PageType.
	PageType	The **PageType** is used to determine how partial results from list data access calls should be grouped while being returned to the client. Choosing the appropriate page type will improve performance.

Member name	Value	Description
TagCount	0	Results are paged from the PI Server by tag count, with a maximum tags per page specified in the PageSize property.
EventCount	2	Results are paged from the PI Server by event count. A new page of data is retrieved when the number of events exceeds the PageSize property. All events for a single tag are retrieved in one page.

Bulk RPC Example - Paging

```
private double GetDataInBulk(AFAttributeList attributes, AFTimeRange timeRange)
{
    // Defined paging strategy
    PIPagingConfiguration pagingStrategy = new PIPagingConfiguration(
        PIPageType.TagCount, 1000);

    // make call.  Return is an Enumerator
    var results = attributes.Data.Summary(
        timeRange, AFSummaryTypes.Total,
        AFCalculationBasis.EventWeighted, AFTimestampCalculation.Auto,
        pagingStrategy);

    // paging will occur behind the scenes as we enumerate
    double total = 0;
    foreach (var result in results)
    {
        AFValue v = result[AFSummaryTypes.Total];
        if (v.IsGood && v.Value is double)
            total += (double)v.Value;
    }

    return total;
}
```

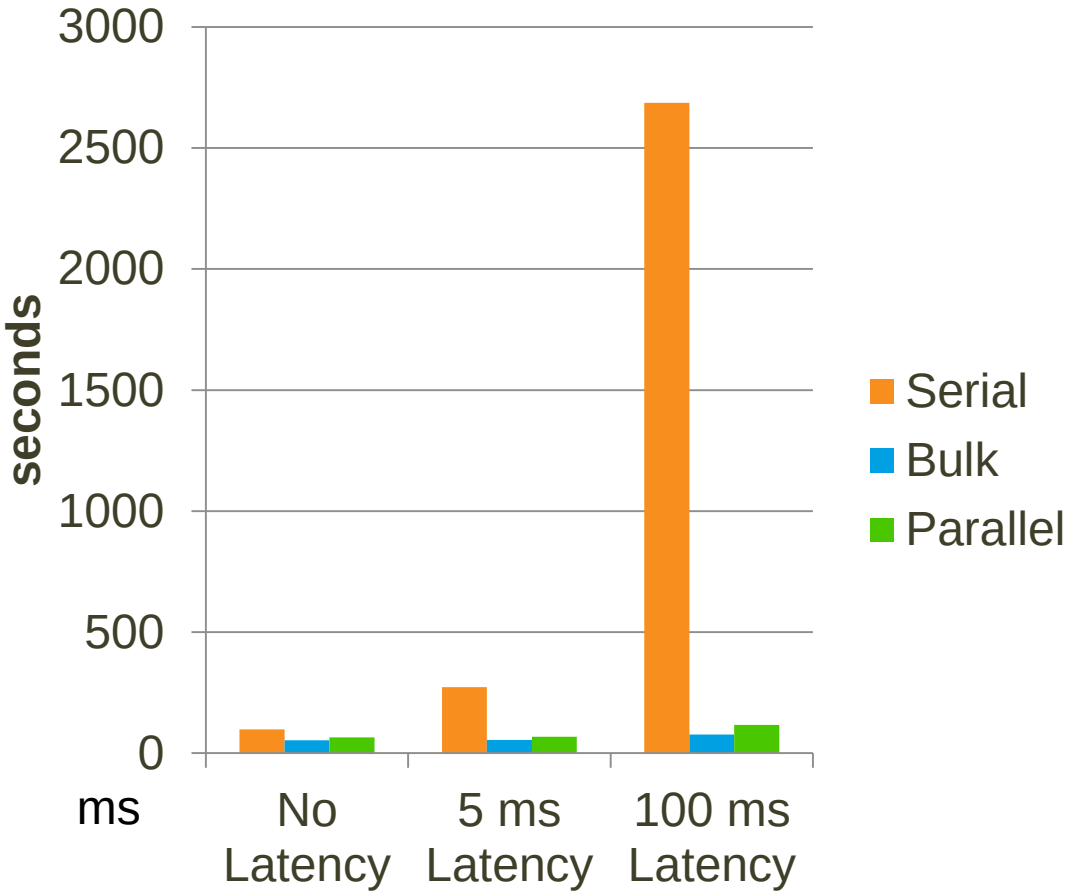
Bulk Performance

RecordedValues

PI Server 2012

25K points

3 million events retrieved



Bulk Performance

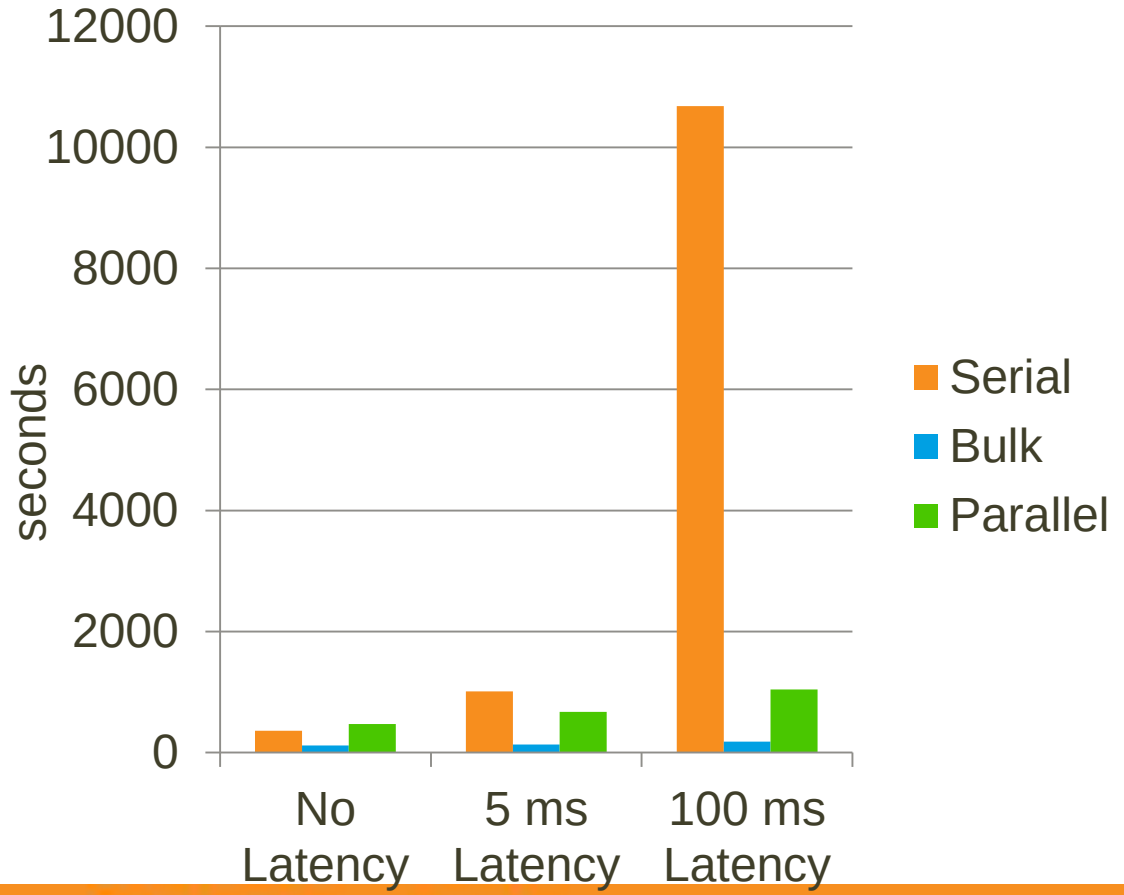
Summaries

PI Server 2012

25K points

1 summary per point

1000 summed events per point

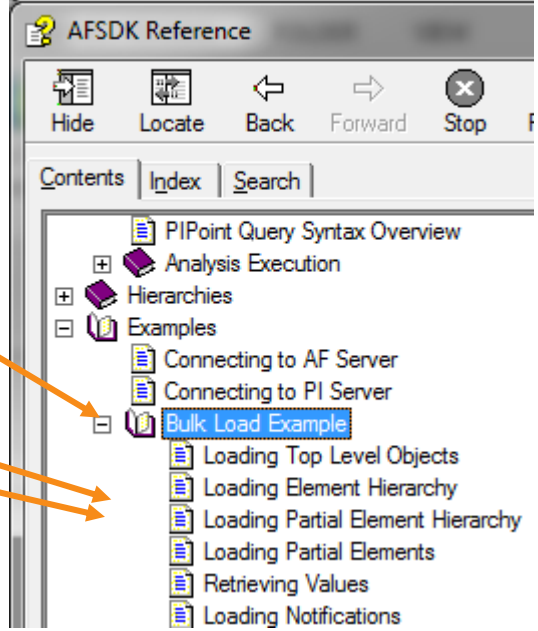


New Bulk Call Behaviors

- Native support on PI Server 2012 and later
- AF SDK will parallelize calls on older Servers
- Non-PI Data Reference Attributes may not use Bulk RPC
- If Filter is not the exact same for all points, call will be done in parallel, not bulk.
- Bulk more predictable than Parallel
- Bulk works better on low-powered clients

Bulk calls to AF Server

- Use Search over Hierarchy
- Load In Bulk
- Load Partial Elements



Caution:

- Recursive hierarchy walking
- For Each on top-level collections
- Complex Configurations (Summary Data References)
- References out-of Element in Data References
- Custom Data References which get own inputs
- Large AF Tables (use Parameterized Tables in AF 2014)

Make Calls In Parallel

- Bulk calls not always possible
  - Too hard to setup in your application
  - Not the same request across points
  - You have a lot of Client side processing
- Utilize idle client Cores
- Example – processing Event Frames

Parallel Example

```
private double GetDataInParallel(AFAttributeList attributes)
{
    double total = 0;
    object monitor = new System.Object();

    // process in parallel.  Each summary is a different time range
    // so no bulk call is available
    Parallel.ForEach(attributes, attribute =>
    {
        AFEventFrame ef = attribute.Element as AFEventFrame;
        var result = attribute.Data.Summary(
            ef.TimeRange, AFSummaryTypes.Total,
            AFCalculationBasis.EventWeighted, AFTimestampCalculation.Auto);

        AFValue v = result[AFSummaryTypes.Total];
        if (v.IsGood && v.Value is double)
        {
            // need to lock when accessing shared "total"
            lock (monitor)
            { total += (double)v.Value; }
        }
    } );
    return total;
}
```

Parallel Cautions

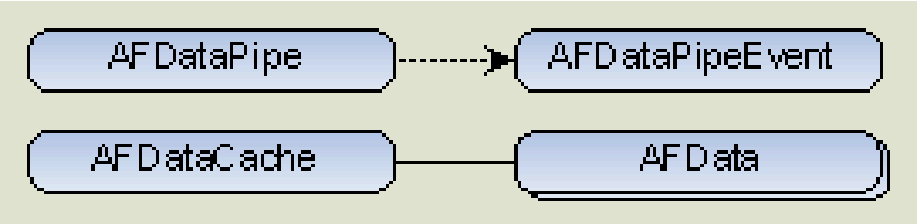
- Lots of small RPCs not as efficient as Bulk
  - For Server, Client, and Network
- Coding is more complex
  - Locking required
- Mileage varies more
  - Differing capabilities on different computers
- You can overstress a system
  - use Tasks and Parallel constructs to reduce stress

Stream Processing

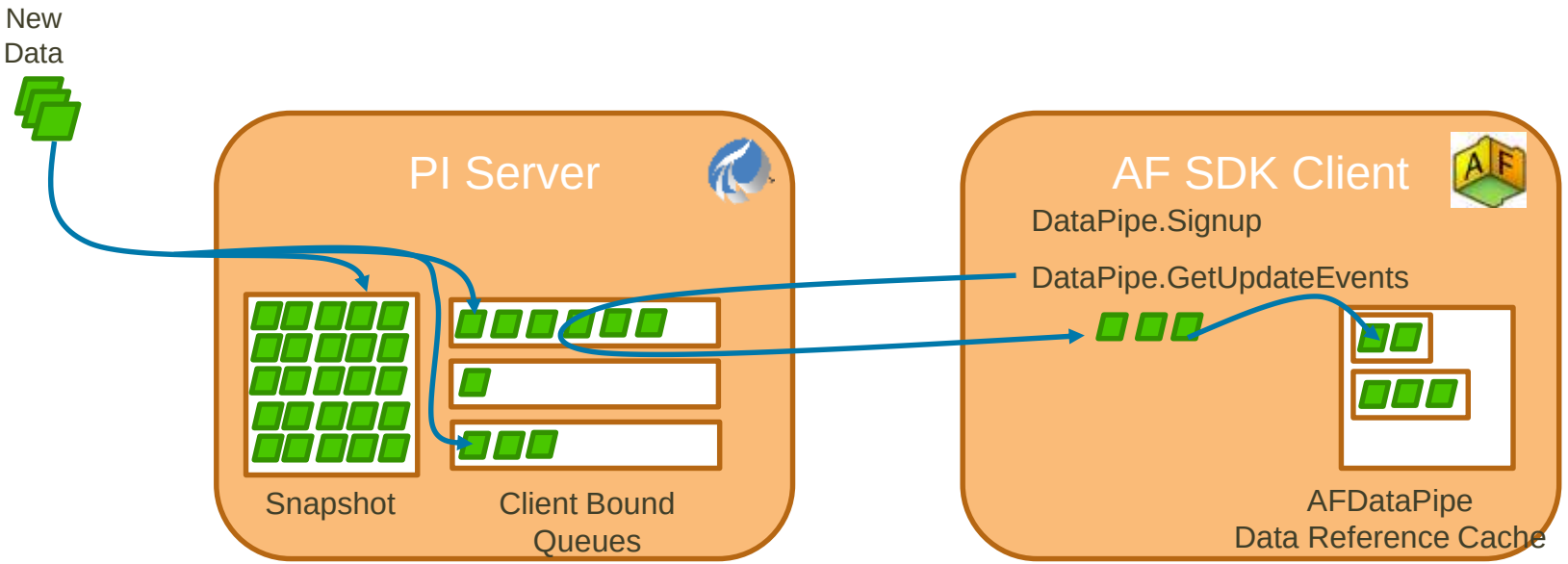
Watching for data flowing into PI and processing only changes

- Support for new Abacus Scheduling Engine

- New AF Classes
  - AFDataPipe
  - AFDataCache



Data Pipe



4 Pipe Types: PI-Snapshot, PI-Archive, PI-Combined (2014), AF (Combined).

AF Data Pipe maintains client side cache of inputs for some calculation based Data References.
PI Data Pipe has no client cache.

Data Pipe

- PIDataPipe released in AF 2012 (2.5) for PIPoints
  Snapshot, Archive. Adding Combined in PI 2014
- AFDataPipe new in AF 2014 (2.6) for all Attributes
- New IObserver Pattern
- It is the Most Efficient way to get real-time data
  - Done in bulk
  - Server only sends Changes of snapshot
  - Server buffers events between client fetch
- Run multiple pipes in parallel

AFDataPipe Example

Signup

```
AFDataPipe _dataPipe;
System.Timers.Timer _timer;
private void Signup(AFAttributeList attributes)
{
    _dataPipe = new AFDataPipe();
    var errors = _dataPipe.AddSignups(attributes);

    if (errors.HasErrors)
    {
        ProcessErrors(errors);
    }
    else
    {
        _timer = new System.Timers.Timer();
        _timer.Interval = 1000;
        _timer.Elapsed += TimerElapsed;
        _timer.Start();
    }
}
```

Process Events

```
private void TimerElapsed(object sender, EventArgs e)
{
    bool moreEvents = true;
    while (moreEvents)
    {
        var results = _dataPipe.GetUpdateEvents(100, out moreEvents);

        foreach (AFDataPipeEvent evt in results.Results)
        {
            ProcessEvent(evt);
        }
    }
    _timer.Start(); // restart the timer.
}
```

AFDataPipe Observer Pattern

Signup

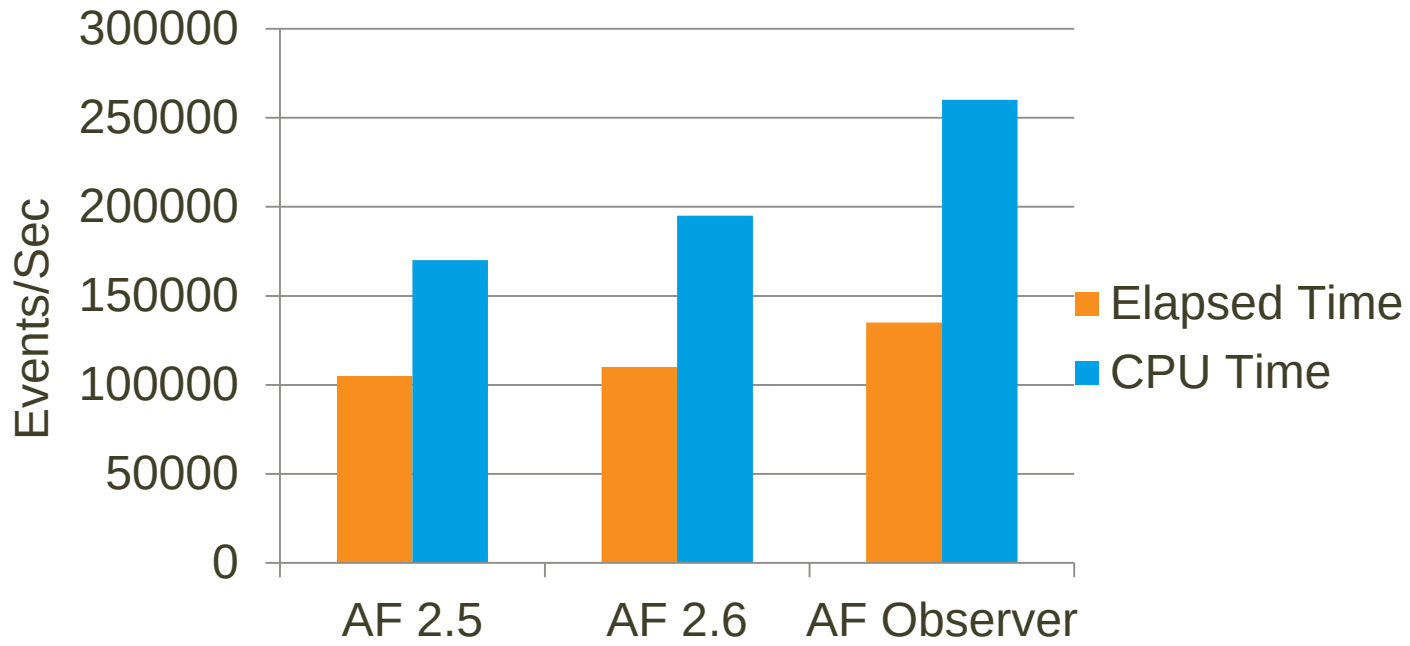
```
EventObserver _observer;
private void SignupObserver(AFAttributeList attributes)
{
    _dataPipe = new AFDataPipe();
    var errors = _dataPipe.AddSignups(attributes);

    _observer = new EventObserver();
    _dataPipe.Subscribe(_observer);
    _timer = new System.Timers.Timer();
    _timer.Interval = 1000;
    _timer.Elapsed += delegate(object o, ElapsedEventArgs e)
        {
            bool moreEvents = true;
            while (moreEvents)
            {
                _dataPipe.GetObserverEvents(100, out moreEvents);
            }
        };
    _timer.AutoReset = true;
    _timer.Start();
}
```

Process Events

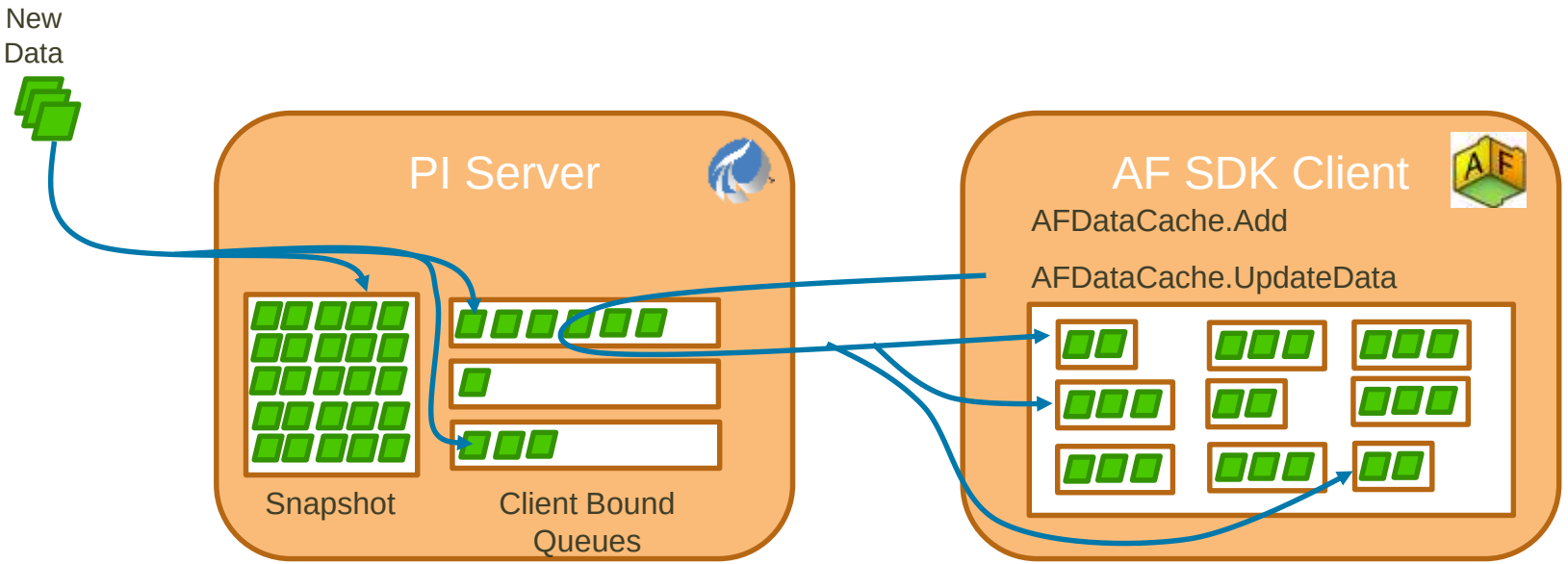
```
public class EventObserver : IObserver<AFDataPipeEvent>
{
    private int _count = 0;
    private bool _isComplete = false;

    #region IObserver<AFDataPipeEvent> Members
    public void OnNext(AFDataPipeEvent evt)
    {
        ProcessEvent(evt);
    }

    public void OnCompleted()
    {
        _isComplete = true;
    }

    public void OnError(Exception error)
    {
    }
```

Observer Pattern removes list creation, improving throughput

Data Pipe – Events per Second/Thread



Client Side Cache



AFDataCache utilizes Data Pipe mechanism for feeding the cache.

AFDataCache maintains cached “AFData” for each attribute. It feeds the cache via UpdateData.

AFDataCache

- Holds Data Client side
- Data can be used multiple times
- Useful when client may not be able to bulk
- Good for Time Synchronizing Data
- When Bulk, Parallel, and Stream are not enough

AFDataCache

Properties

	Name	Description
	CacheTimeSpan	The minimum amount of time series data to be kept for each AFAttribute in the cache, as measured by time span.
	MaxCacheEventsPerAttribute	The maximum number of cache events to be kept for each AFAttribute in the cache.
	MinCacheEventsPerAttribute	The minimum number of cache events to be kept for each AFAttribute in the cache.

Methods

	Name	Description
	Add	Adds a list of AFAttribute objects to be managed by AFDataCache. The AFAttributes are reference counted. Adding AFAttribute that is already in the AFDataCache will increment the reference count and not generate error. Adding the same AFAttribute multiple times will require the same number of Remove call to actually remove the AFAttribute from the AFDataCache.
	TryGetItem	Returns the cache enabled AFData object for a given AFAttribute.
	UpdateData()	Update the time series cache for the attributes managed by by this AFDataCache.

AFDataCache Example

Setup

```
AFDataCache _dataCache;
private void Setup(AFAttributeList attributes)
{
    _dataCache = new AFDataCache(attributes.Count);
    _dataCache.MaxCacheEventsPerAttribute = 10;
    _dataCache.MinCacheEventsPerAttribute = 2;
    _dataCache.CacheTimeSpan = TimeSpan.FromMinutes(2);
    _dataCache.Add(attributes);

    _timer = new System.Timers.Timer();
    _timer.Interval = 1000;
    _timer.Elapsed += delegate(object o, ElapsedEventArgs e)
      { _dataCache.UpdateData(); };
    _timer.AutoReset = true;
    _timer.Start();
}
```

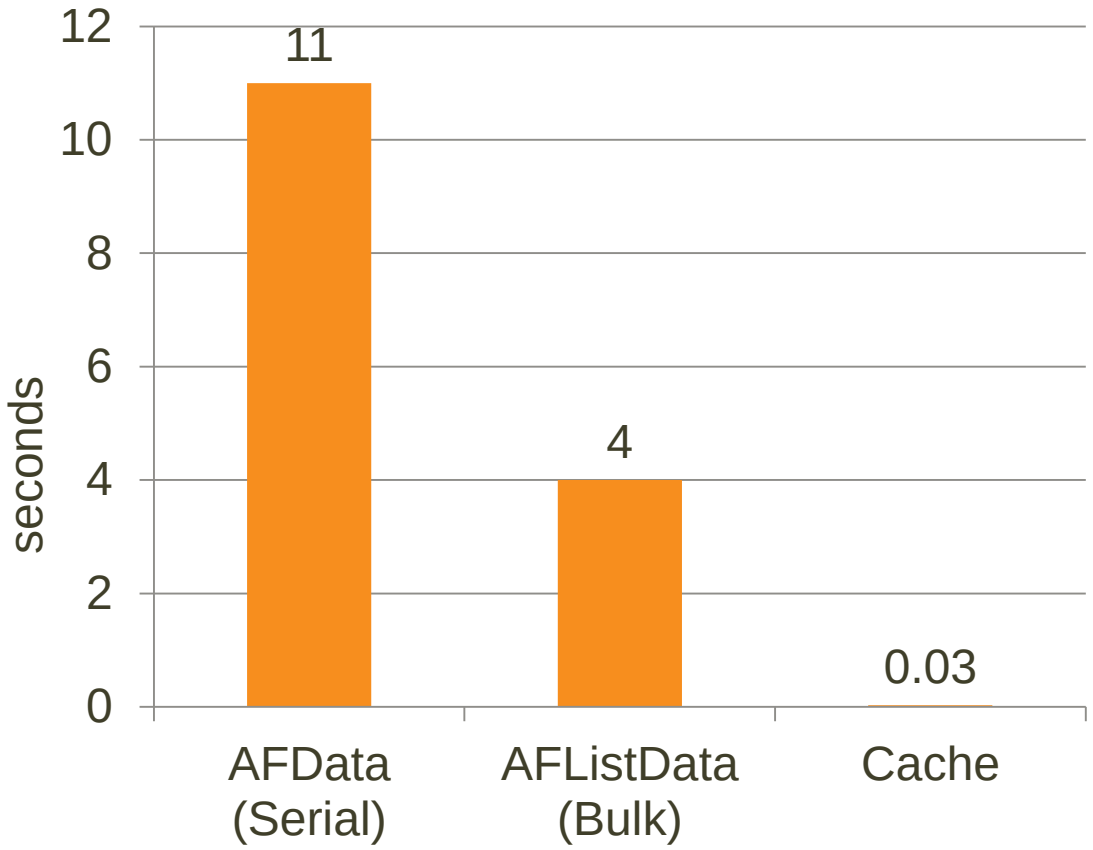
Use

```
private AFValue UseCache(AFAttribute attribute, AFTime time)
{
    AFData afdata;
    // retrieve the cache aware AFData object
    if (!_dataCache.TryGetItem(attribute, out afdata))
    {
        afdata = attribute.Data; // no cache.
    }

    return afdata.InterpolatedValue(time, null);
}
```

AFDataCache Performance

InterpolatedValue

10,000 Attributes



AFDataCache Cautions

- Potential for Stale Data
- High Memory Usage
- Not all AFData calls are cache-aware
- Small practical time range

Pre-Calculate

- **Sometimes, there is just too much data to calculate on demand.**
- **Use Abacus to pre-calculate instead of Data References**
- **Resultant Data queries can be orders of magnitude faster**

Pre-Calculate - Tradeoffs

- Late Arriving Data
- Back-Calculations
- Fidelity of Data
- Additional Tags

"You can't ride two horses with one butt"
Russian Proverb.

Pre-Calculate: Point Path Resolution

PI System Explorer shows simplified name:

\\myPIServer1\Tank101.Temperature

Actual stored string may differ:

- Calculated from the template:

  \\%@myServer%\%Element%.%Attribute%

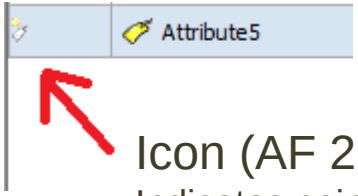
- Unresolved:

  \\myPIServer1\Tank101.Temperature

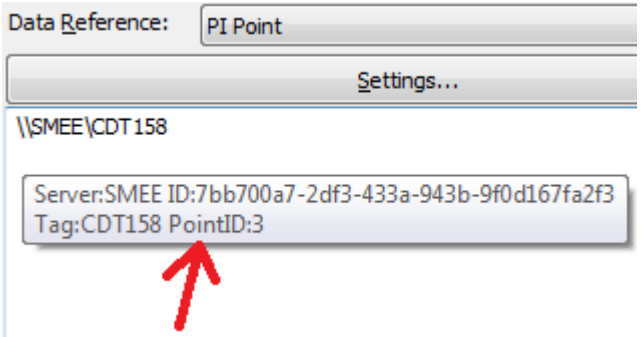
- Resolved:

  \\myPIServer1?fa2958bc-b5ab-4686-a5f2-16155eb0ab71\Tank101.Temperature?10891

Point Path Resolution- How do you Know?



Icon (AF 2.6 / 2014)
Indicates point could be auto created or resolved



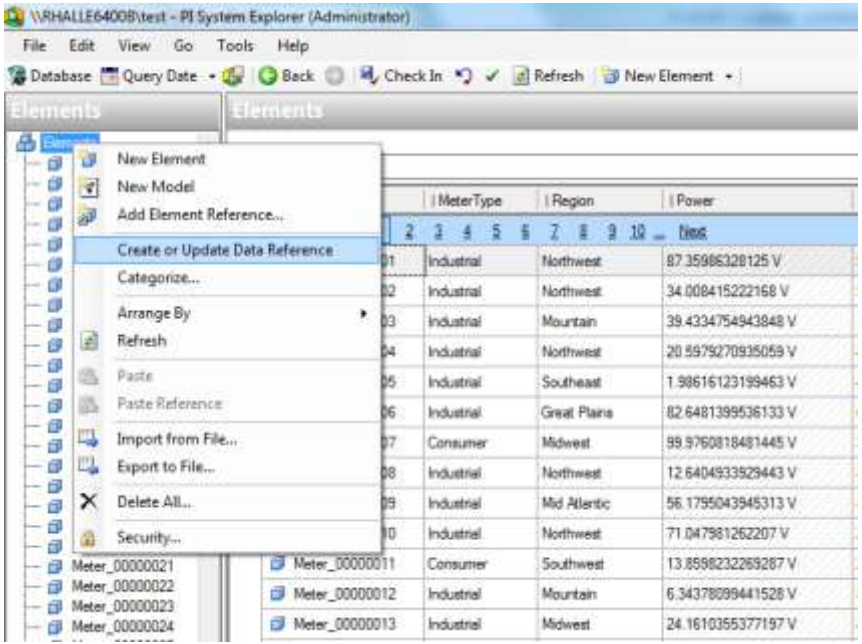
Tooltip (2.6 / 2014)

AFExport (all AF Versions)

```
C:\Program Files\PIPC\AF\AFExport "\\.\ MyDatabase \ MyElement | Temperature" | findstr ConfigString

<ConfigString>\\myPIServer?7bb700a7-2df3-433a-943b-9f0d167fa2f3\CDT158?3</ConfigString>
```

Point Path Resolution

Resolve in PSE



New Command Line Tool

```
C:\Program Files\PIPC\AF\
AFUpdatePluginConfiguration /Resolve
"\\.\MyDatabase"
```

Time = 26.6 s

Diagnosing Performance Tools

New in AF 2014

- **RPC Metrics**

```
AFRpcMetric[] newClientMetrics = myPISystem.GetClientRpcMetrics();
```

  RPC metrics to af server, client measurements

- **RPC Metrics**

```
AFRpcMetric[] newServerMetrics = myPISystem.GetRpcMetrics();
```

  RPC metrics to af server, server measurements

New in AF 2014

- **SQL Tracing**

```
>AFDiag /STP:2013/01/01
```

  Enable and report through AFDiag utility

New in AF 2014

- **AFGetTrace**

```
>AFGetTrace /EnableAF /EnablePI
```

  Client call level trace detail

- **PI System Management**

  RPC metrics to PI server, server measurements

Summary
Strategies for Optimizing SDK Applications

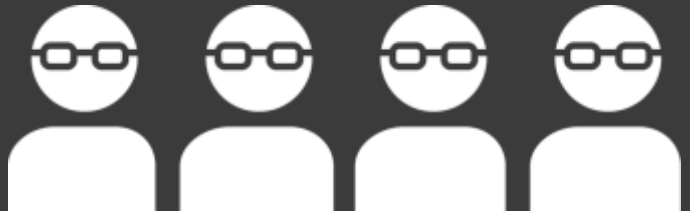
- **Bulk**
- **Parallel**
- **Stream (Data Pipe)**
- **Cache on Client**
- **Pre-Calculate**
- **Diagnose Performance Bottlenecks**

Remember – at large scale – everything matters

Optimizing Server Side

“You can’t have more than there is”
***Paul Combellick*.**

Optimizing your AF Server



PI AF Server Topics

- Deployment options for PI AF 2.5 (PI AF 2012)
- New or Improved in PI AF 2.6 (PI AF 2014)

Deployment

- Recovery Model
- SQL Server Editions / Versions
- Database File Sizing
- Memory Sizing
- Disk Configuration

Deployment
SQL Server Recovery Models

- **Simple** – default for a new AF installation
  - Transaction log does not grow.
  - Best for test systems or any system that you can rebuild the data.
    - ALTER DATABASE [PIFD] SET RECOVERY SIMPLE;
- **Full**
  - Transaction log grows until log backup.
  - Best practice for production systems.
  - Can recover post-backup data changes after data disk failure.
    - ALTER DATABASE [PIFD] SET RECOVERY FULL;

  - http://msdn.microsoft.com/en-us/library/ms189275(v=sql.110).aspx/css

SQL Server Recovery Models Best Practice

- Use Simple Recovery for Dev / Test systems.
- Use Full Recovery for Production systems.

Deployment
SQL Server Versions / Editions

Versions

- PI AF 2.5 supports SQL Server 2005 – 2012
- PI AF 2.6 supports SQL Server 2008 – 2012 (Testing with SQL Server 2014 CTP2 and SQL Azure)

Editions

- Express
  - Severely limited: 1GB RAM, 1 CPU, data + log size limited to 10GB, No job scheduler (SQL Agent),
  - Effectively limited to a few tens of thousands of AF Objects
- Standard
  - Max 16 cores, 64GB RAM
  - Clustering , mirroring, AF Collective primary
- Enterprise
  - RAM & Max Cores limited by Windows
  - Clustering, mirroring, AF Collective primary, Always-On
  - Change Data Capture – PI AF2.6 Audit trail
- Developer – included in MSDN license

```
SELECT @@SERVERNAME, @@VERSION ;
```

SQL Server Versions / Editions Best Practice

- Small Systems (< 1.5 GB data, a few users)
  - SQL Express

- Require high availability, compressed backups, large amount of RAM or CPU, PI AF 2014 Audit trail:
  - SQL Server 2012 Enterprise Edition

Deployment
PIFD Database File Sizing

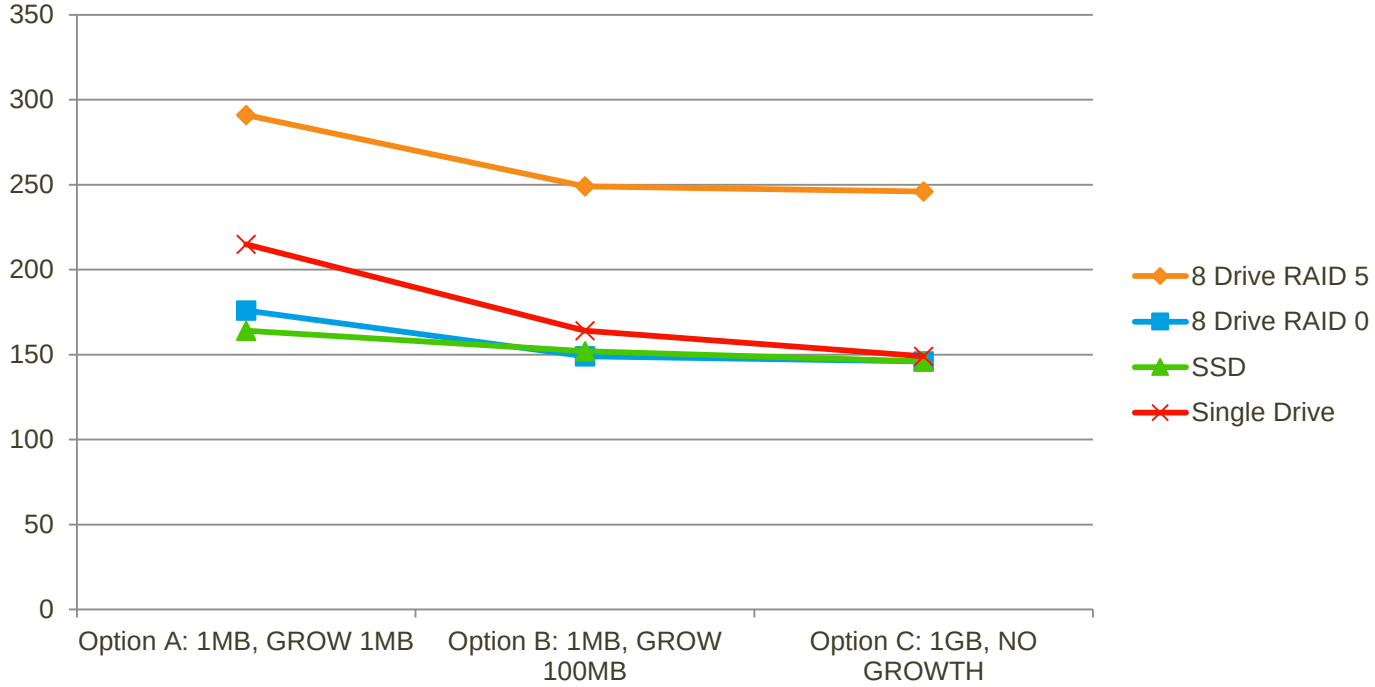
- Auto growth of database files adversely affects write performance except on SSD.

Deployment
PIFD Database File Sizing

- Auto growth of database files adversely affects write performance.
- PI AF 2.5: Files start at 1MB and grow by 1MB
    - A deployed system can be modified using SSMS
- PI AF 2.6: Files start at 1MB and grow by 100MB

Deployment
PIFD Database File Sizing

Time (seconds) to insert 50K AFElements



PI AF 2.5
SQL Express 2012

PIFD Database File Sizing Best Practice

- Pre-allocate file size, except on SSD.
- Disable AUTO_SHRINK.
- Don’t shrink database files.

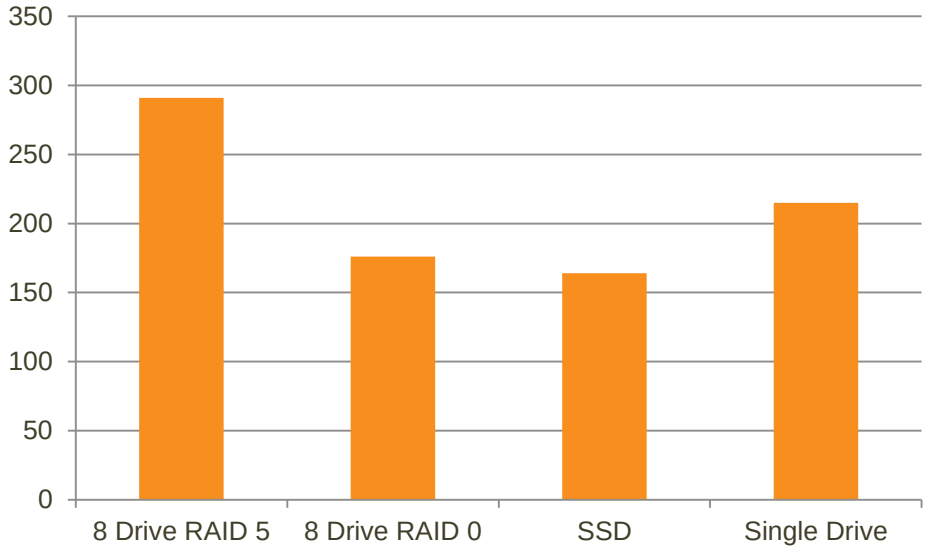
Deployment
Memory Sizing

- Quantity of RAM allocated to SQL Server can affect read and write performance.

- Generally, want RAM allocated to SQL Server Instance to exceed 60% of the size of the database(s).

Deployment
Disk Configuration

Time (seconds) to insert 50K AFElements



PI AF 2.5
SQL Express 2012

Test Platform: Dell Precision T7600

32 GB RAM
24 logical CPU
256GB SSD – 40K IOPS
1TB drive

Windows 8
SQL 2012 Dev
SQL Express 2012

15 SAS Drives 146 GB
4.5K IOPS

Disk Configuration Best Practice

- Choose drive configuration with sufficient IOPs, bandwidth, storage capacity, and availability.

Deployment
High Availability Options

	SQL Express	SQL Cluster	SQL Mirror	AF Collective	SQL Always-On
HA Reads	No	Yes	Yes	Yes	Yes
HA Writes	No	Yes	Yes	No	Yes
Quick Failover	NA	No	Yes	Yes	Yes
Number of Data Copies	1	1	2	2+	2-4

High Availability Options
Best Practice

- 2 or more AF Servers, load balanced

- SQL Server 2012 Always-On
  - New in SQL 2012, Mirroring ++
  - PI AF 2.5 compatibility white paper available.

New or Improved in PI AF 2.6

- Finding Elements By Path
- Insert Performance
- Hierarchical Template Performance
- Maintenance Job
- AF Sql Trace

Improved in PI AF 2.6

Collapse All Code: All Members: Show All

AFSDK Reference

AFElement.FindElementsByPath Method

AFElement Class See Also Send Feedback

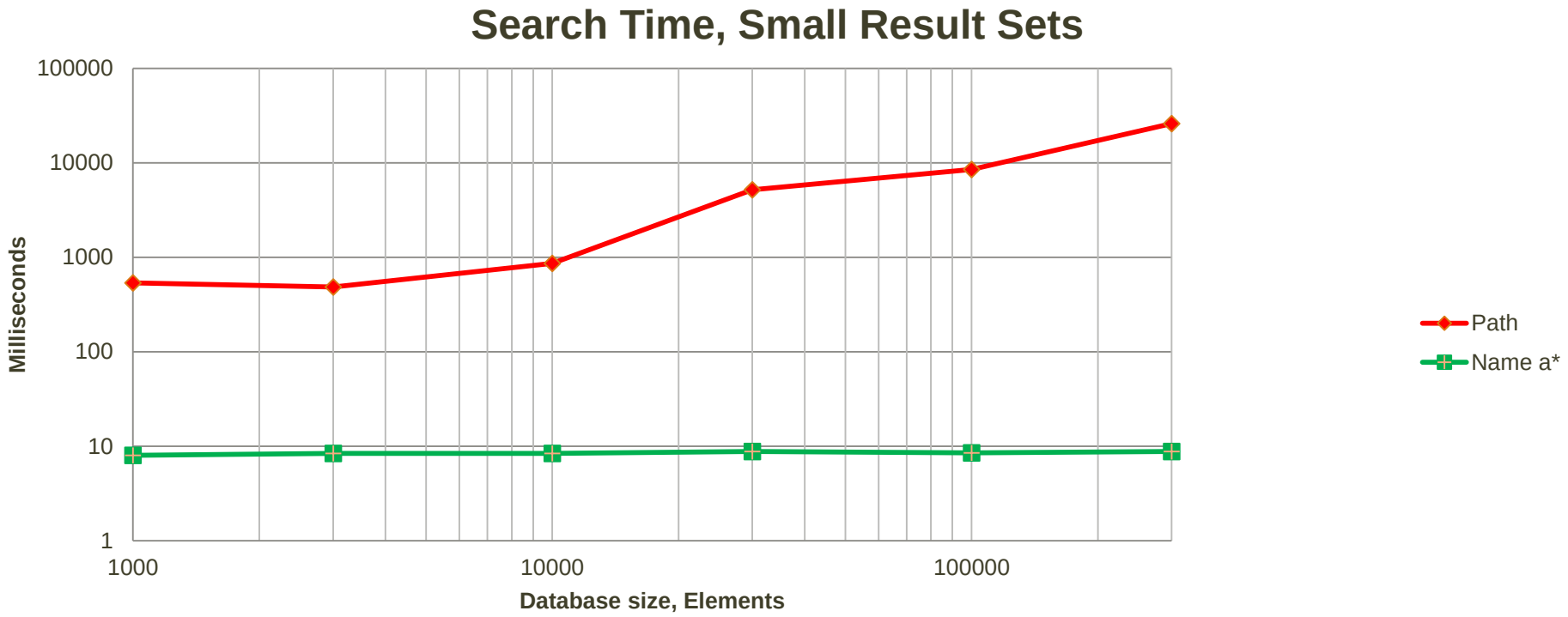
Retrieves the AFElement objects identified by the path strings.

Overload List

	Name	Description
S	FindElementsByPath(IEnumerable (String), AFObject)	Retrieves the AFElement objects identified by the path strings as keyed results.
S	FindElementsByPath(IEnumerable (String), AFObject, IDictionary (String, String))	Retrieves the AFElement objects identified by the path strings as a list.

\\AFServerName\AFDatabaseName\Meters\AcmeHeadEnd00001006\Task1

PI AF 2.5 Searching Metrics

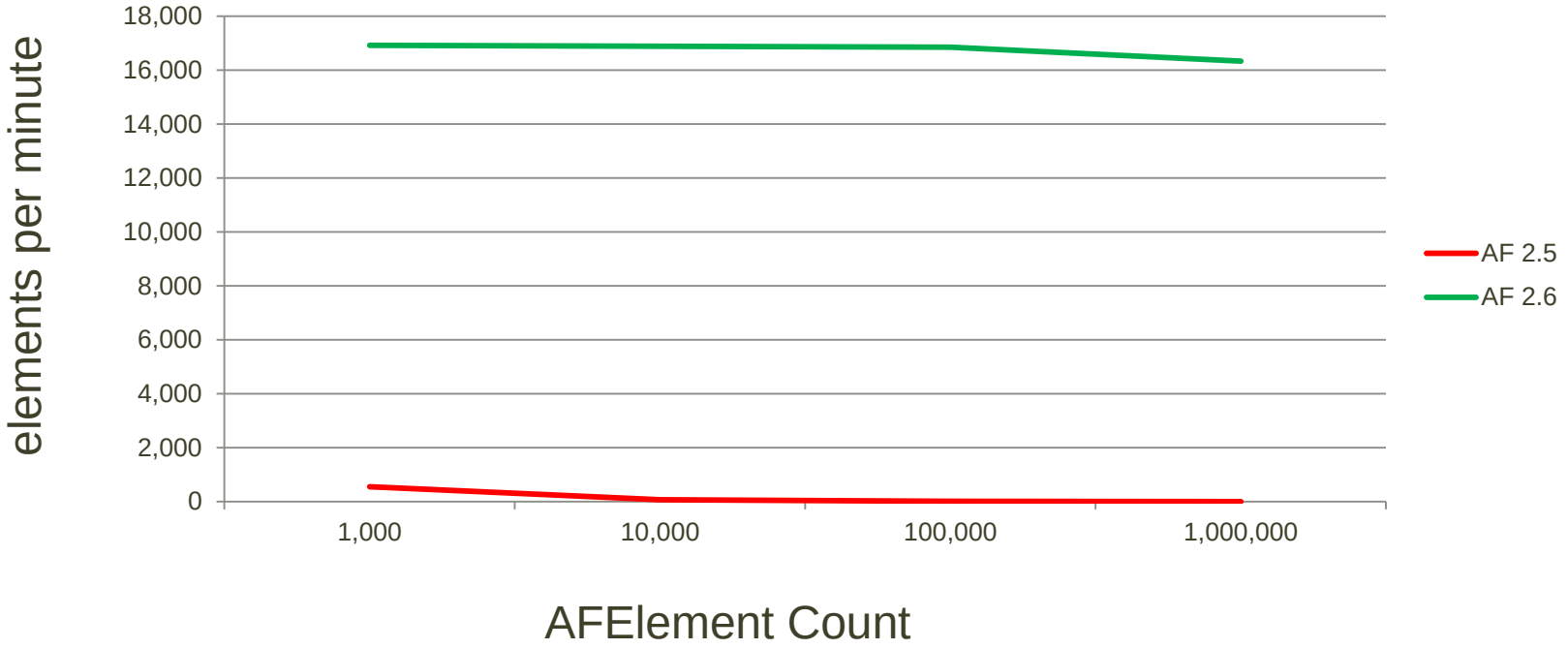


AFElement.FindElementsByPath ( )

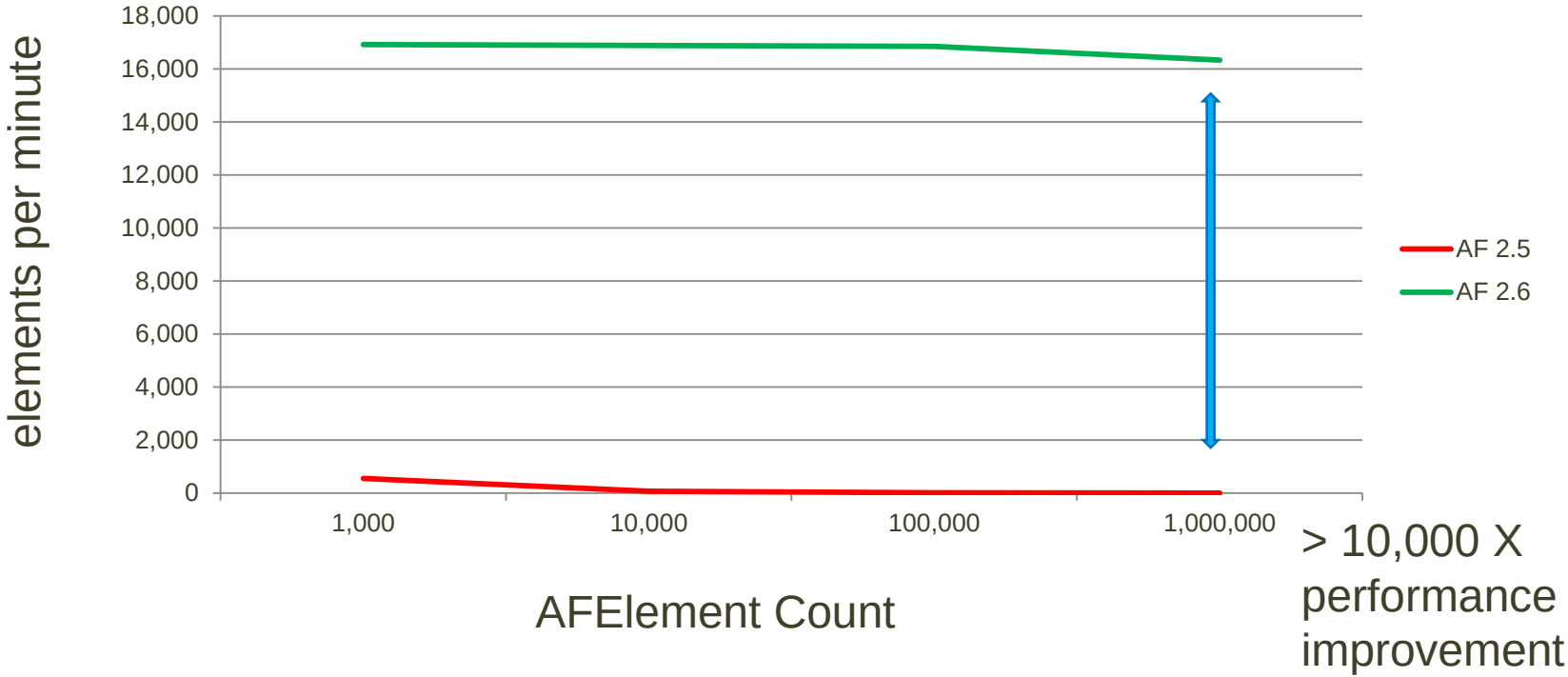
- FindElementsByPath() is used by several client tools.
- Too slow, especially > 10K elements.
- If more than a few concurrent requests, response slows to a crawl, SQL Server CPU -> 100%.

- Can we improve performance?
- Can we improve concurrency?
- What are the tradeoffs?

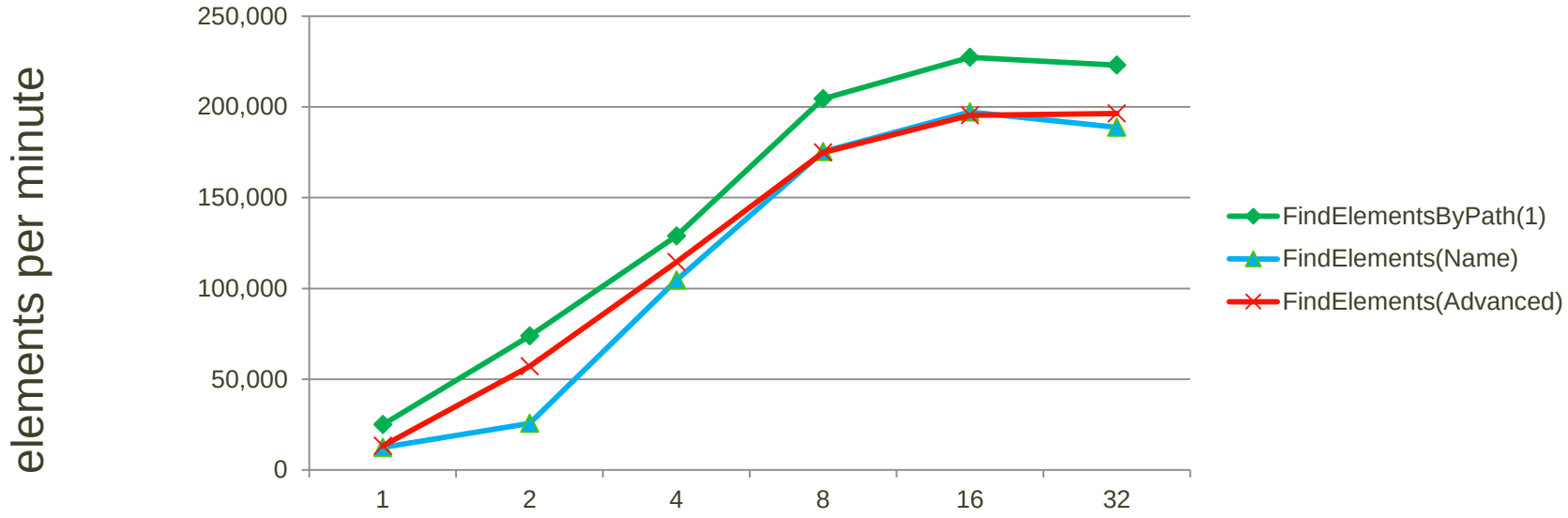
AFElement.FindElementsByPath ( ) Performance - AF2.5 vs AF2.6



AFElement.FindElementsByPath ( ) Performance - AF2.5 vs AF2.6



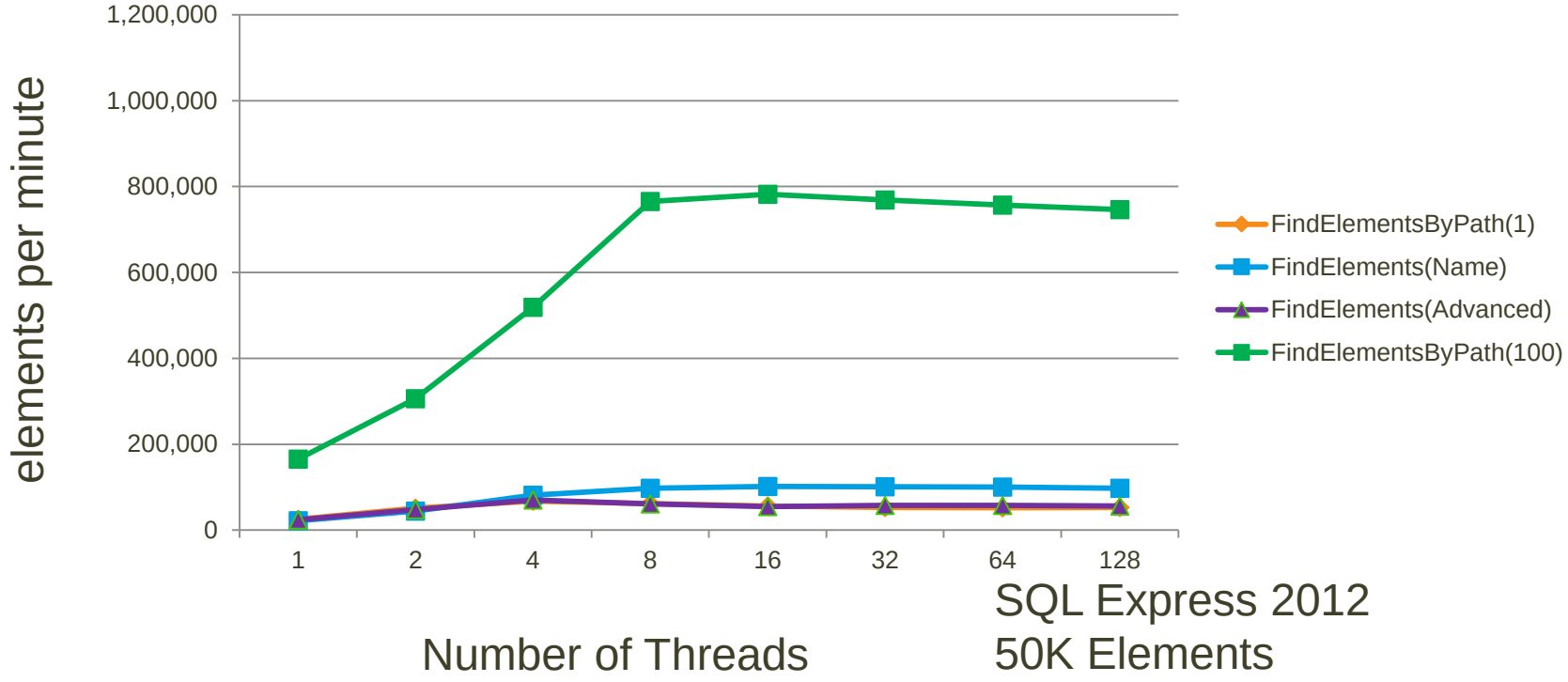
PI AF 2.6 FindElements…( ) Concurrent Requests



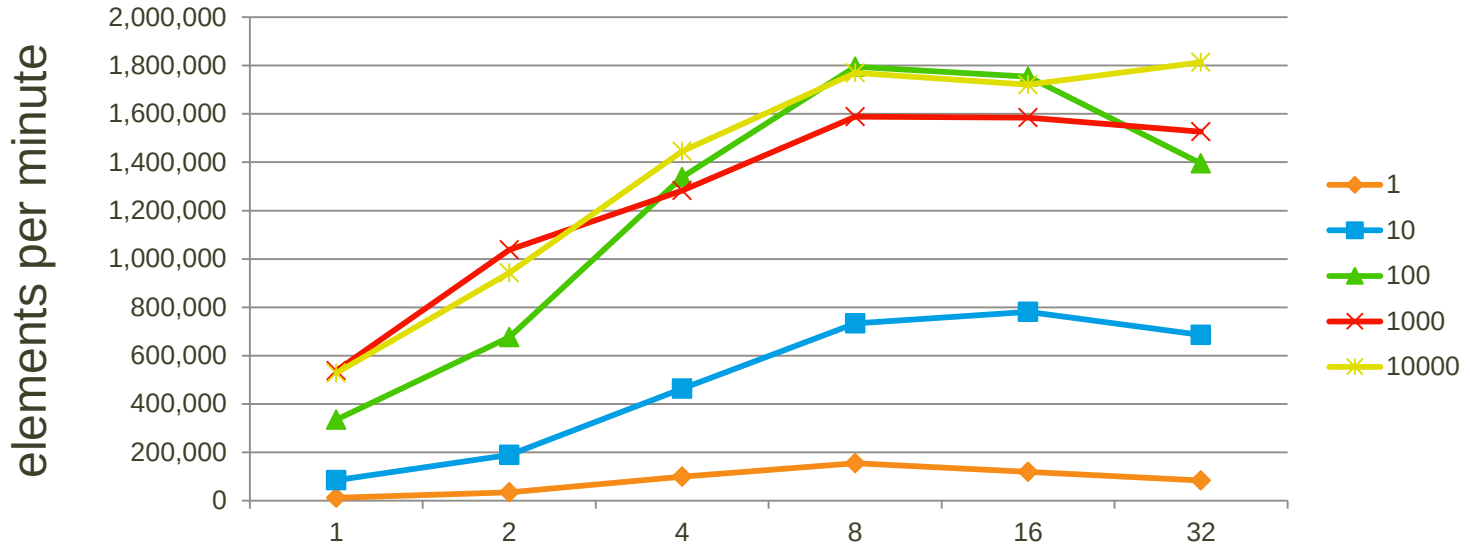
Number of Threads

SQL Express 2012
50K Elements

PI AF 2.6 FindElements…( ) Concurrent Requests

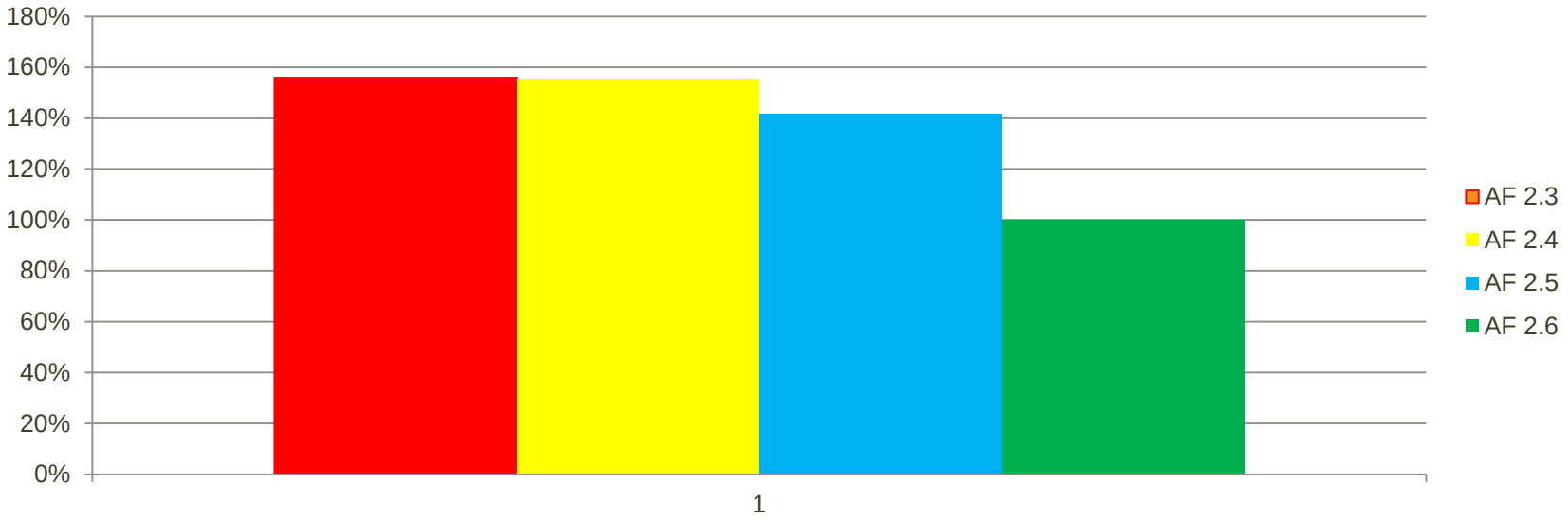


PI AF 2.6 FindElementsByPath ( ) Concurrent Requests – Batch Size



SQL Server 2012 Dev Edition
32GB RAM
6 million Elements

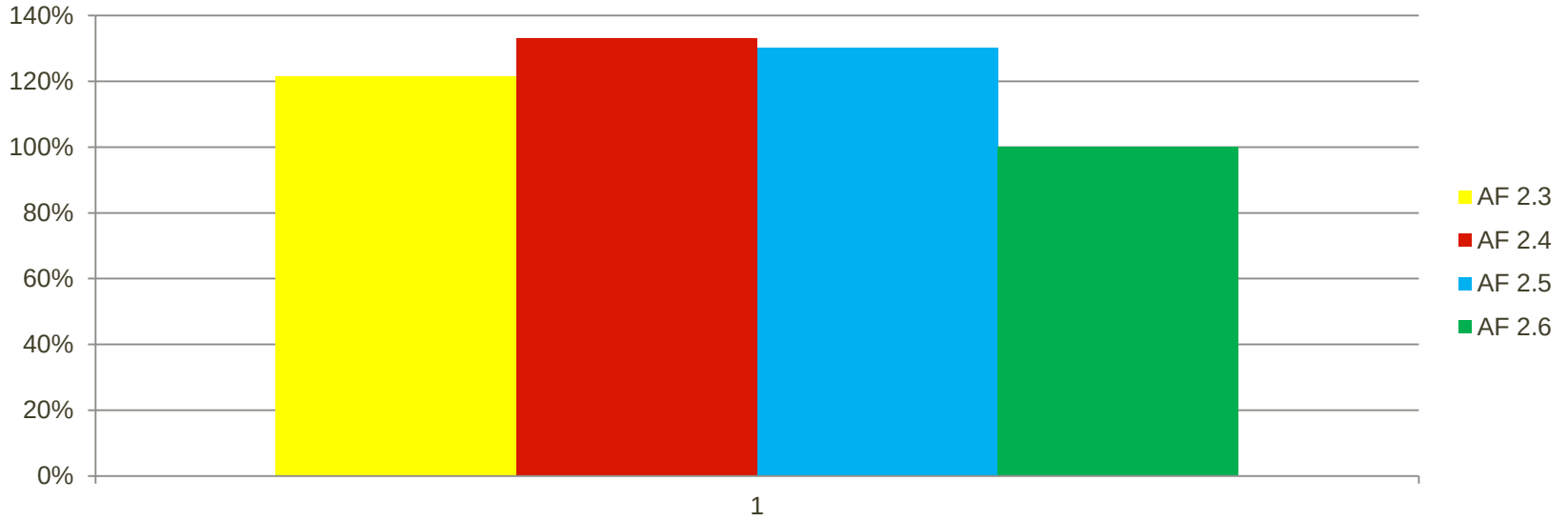
Improved in PI AF 2.6
Insert Performance



AFElement Insert Time relative to PI AF2.6

Single Drive
SQL 2012 Dev

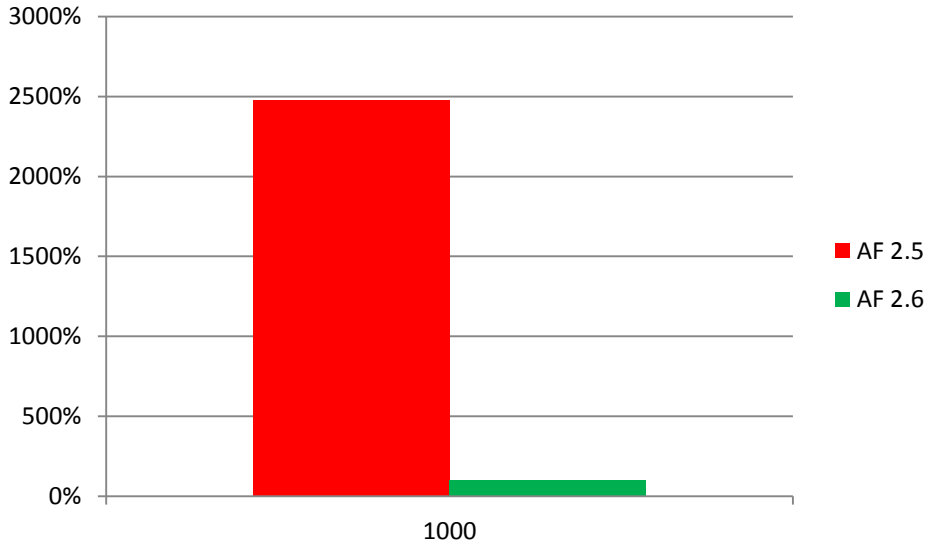
Improved in PI AF 2.6
Insert Performance



Insert Time relative to AF2.6

SSD
SQL 2012 Dev

Improved in PI AF 2.6
Hierarchical Template Performance



FindElementAttributes( ): Elapsed Time relative to AF 2.6

Searching Guidelines

- Best:
  - Path Searches
  - Indexed Attribute Values
    - Even if it means changing the AF Model
  - Inherently Indexed Properties
    - Name, UniqueID
- Good:
  - Other Properties
    - Description, Category, Template
  - Attribute Values (not indexed)
  - Attribute Searches
- Caution:
  - Searches with large results sets (Limited by Bandwidth)
  - Complex Criteria (more joins = slower performance)

New for PI AF 2.6
Maintenance Job

- Re-Indexing
- Statistics Maintenance
- Rebuild Path Cache
- Delete Orphaned Elements
- Delete Orphaned Files, AFDatabases

New for PI AF 2.6
Monitoring Performance: AF SQL Trace

- Enable: afdiag /ST+
- Disable: afdiag /ST-
- Delete: afdiag /ST0
- Summary: afdiag /STP

Key Performance Points – PI AF Server

- Add RAM to SQL Server.
- Use drives with adequate IO / storage capacity.
- Pre-allocate database files.
- Weekly database Maintenance.
- SQL Server Always-On (Mirroring ++).

Chris Manhard

PI AF Engineering
Group Lead

OSIsoft, LLC
cmanhard@osisoft.com

Paul Combellick

PI AF Server
Principal Engineer

OSIsoft, LLC
paulc@osisoft.com

Please don’t forget to…

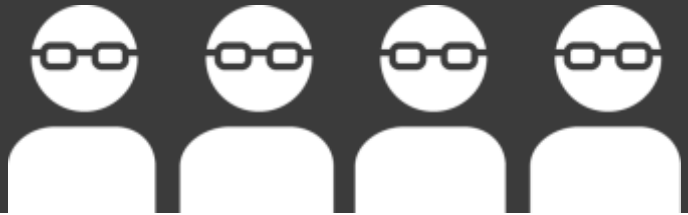
Complete the online survey for this session

eventmobi.com/vcampus13

Share with your friends

#VCL13

THANK YOU



Brought to you by OSIsoft.