

# **Pan-STARRS PS1 Published Science Products Subsystem**

**J. N. Heasley**

*Pan-STARRS Project, Institute for Astronomy, University of Hawaii*

**Robert E. Eek, Wayne L. Smith, Julie A. Rosen**

*Science Applications International Corporation*

**The Pan-STARRS team**

*Pan-STARRS Project, Institute for Astronomy, University of Hawaii*

## **Abstract**

This paper describes the requirements and design of the Pan-STARRS PS1 Published Science Products Subsystem (PSPS) that constitutes the primary distribution tool for the very large amount of science data products produced by the Pan-STARRS PS1 prototype telescope. The data management challenges are identified in terms of stressing characteristics: dynamic, fast, spatial, and large; these are countered by mitigating characteristics: simple and lenient. This combination of characteristics is not only distinctly more demanding than traditional survey astronomy data managers, but lies at the boundaries of current commercially available data management technology. The requirements imposed on the PSPS result in devising a design strategy at the boundaries of currently available data management technology. In particular, we describe the capabilities and characteristics of the four main PS1 PSPS components: the Web-Based Interface (WBI), the Data Retrieval Layer (DRL), the Object Data Manager (ODM), and the Solar System Data Manager (SSDM). Potential architectural strategies are examined in the context of the stressing and mitigating characteristics with the conclusion that the ODM should follow an architectural concept that emphasizes the pooling of application, processing, and storage resources. The PS1 PSPS is specifically designed to support the PS1 science mission (see Chambers *et al.*, these proceedings) while at the same time providing substantial design direction for a future PSPS component of the final PS4 Pan-STARRS. Finally, the limitations and possible scalability of the PS1 design relative to PS4 are discussed.

## **1 Introduction**

The Published Science Products Subsystem (PSPS) within Pan-STARRS serves as the archive and access point for all data products published by the Pan-STARRS system; see Figure 1. The PSPS will receive publications from other Pan-STARRS components, including the Image Processing Pipeline (IPP), the Moving Object Processing System (MOPS), and potential future science processing clients. The PSPS is a multi-tiered system, consisting of the following major components:

- Web-Based Interface (WBI), which provides end-user access to the published products;
- Data Retrieval Layer (DRL), which integrates all data management components and provides Web Service (i.e., software) access to the published products; and
- Data Management Components (DM Components), which manage and provide access to specific collections of published products.

The term “published” is important in this context, and needs explicit mention. PSPS will not provide access to *all* of the data produced or managed within Pan-STARRS, rather only those data products that have been determined to be suitable for publication. It is expected that publication suitability will be determined by a defined data-processing stage, rather than an individual assessment of any particular data product. In other words, data from intermediate steps in a processing pipeline are not likely to be published, and therefore will not be archived within the PSPS.

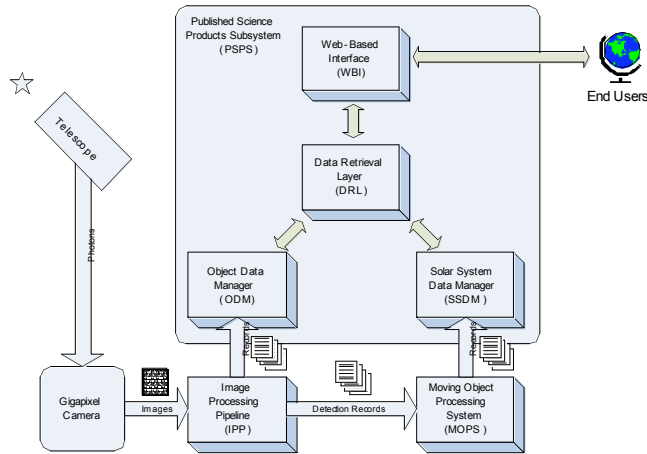


Figure 1. Components of the Pan-STARRS System

Due to the technical risks in the development and deployment of the full four-telescope Pan-STARRS, the project is pursuing a reduced-scale prototype to investigate the higher risk components of the full system. This operational prototype is called PS1, and its PSPS development focuses on two data management components:

- The Object Data Manager (ODM), which will manage catalogs of detections and derived celestial objects from the IPP, and
- The Solar System Data Manager (SSDM), which will manage catalogs of high-quality orbit determinations and their attributed detections.

Other data management components, required for full Pan-STARRS development, were intentionally excluded from the PS1 PSPS in an effort to focus on the high-risk technical issues and reduce financial cost.

The ODM is viewed as the high-risk data management component due to the data entity counts, ingest rates and access patterns (both end-user access and internal processing access). The SSDM is not considered a high-risk component, since it is relatively small and simple. However, implementation of the SSDM component is expected to provide high value for future science with minimal cost. Additionally, implementation of this second data manager component ensures appropriate attention to DRL integration issues.

This paper is intended to provide a concise description of the ODM issues, possible ODM architectures, and the prototype design we have chosen to implement.

## 2 ODM Published Products

The ODM will archive and provide access to data products that are published from the Image Processing Pipeline (IPP). To understand these products and how they interrelate, we provide a brief introduction to the IPP's published products.

## 2.1 IPP Products

The PS1 ODM will administer published products that are directly produced by the IPP. These products and their attributes are the input data for the ODM.<sup>1</sup> As this paper is written, the authors understand that the IPP will publish the following products to PSPS:

- **Simplified Frame Metadata:** A record for each P2 frame, each P4 $\Delta$  frame, and possibly one for each Cumulative Sky slice. Important attributes are the sky position, filter and time of the observation (frame).
- **P2 Detection Records:** A record for each celestial object found in each P2 frame. These detection records will arrive from the IPP already associated with their respective frame record. Important attributes of these detection records include the sky position, magnitude (brightness), and associated errors. Additionally, the IPP may provide an identifier for the celestial object itself.
- **Cumulative Detection Records:** A record for each celestial object found in each Cumulative Sky slice. Important attributes of these cumulative detection records are the sky position, magnitude, associated errors, and shape information, as appropriate.
- **5 $\sigma$  Delta Detection Records:** A record for each difference seen at the 5 $\sigma$  level in each P4 $\Delta$  frame. These 5 $\sigma$  delta detection records will arrive associated with their respective frame record. Many of these records will not correspond to any known celestial object. Important attributes of these 5 $\sigma$  delta detection records are the sky position, magnitude, and associated errors.
- **3 $\sigma$  Delta Detection “Sets”:** The collection of all 3 $\sigma$  delta detection records for each P4 $\Delta$  frame. The limited use of these records, when compared with their large numbers, makes it acceptable to publish them in a single collection; that is, 3 $\sigma$  delta detection records will not be published as individual entities.

Except for the 3 $\sigma$  Delta Detection “sets,” all of these products are simple, relatively small records. This fact is noted here to preface the subsequent discussion of the ODM’s challenges in which the authors focus on the number of objects and their required processing throughput, but not the length (or size) of the individual records.

## 2.2 Derived Products

The IPP is responsible for most of the computational analysis of the gathered data. This said, the ODM will derive at least one published product based on the information provided by the IPP:

- **Celestial Object Records:** A record for each celestial object in the sky. The properties of each object will be derived from the set of all detection records associated with the object. Important attributes of these celestial object records include sky position, average magnitude in each filter, proper motion (if appropriate), photometric redshift (if appropriate), and errors or statistics related to the derived properties.

This derived product will be generated from the set of all detections of the same celestial object. Every celestial object record will be associated with at least one detection record (of the various forms produced by the IPP), and each detection record will be associated with one (and only one) celestial object record.

The derived attributes of the celestial object record fall into three general categories:

1. Statistical attributes that apply to all celestial objects and require access to the full time-history of detections for that object (e.g., average magnitudes or variances);
2. Attributes that apply to only some of the possible celestial objects, and require access to the full time-history of detections for those objects (e.g., proper motion values); and
3. Attributes that can be calculated from the statistical properties of the objects themselves, without access to the time-history of detections (e.g., colors, photometric redshifts).

It is possible that these various attributes can be generated on different schedules, as appropriate.

---

<sup>1</sup> The detailed attributes of these records are not discussed in this paper, but the attributes that indicate important relationships to other entities managed by the ODM are highlighted.

**Table 1. PS1 Published Entities Estimate**

<b>Record Type</b>	<b>1-Year Total</b>	<b>Publish Action</b>	<b>Average Rate</b>
Frame Metadata	400,000	Add new record	1000 / day
P2 Detections	40 billion	Add new record	100 million / day
Cumulative Detections	40 billion	Add new record	100 million / day
5 $\sigma$ Delta Detections	3 billion	Add new record	10 million / day
3 $\sigma$ Delta Detection Sets	200,000	Add new record	500 / day
Celestial Objects*	7 billion	Update existing record	100 million / day

\* See the discussion on celestial object maintenance in Section 3.5 for an explanation of why this number is not simply the sum of the detection rates.

As more detections of a celestial object occur, the calculated sky position of the object changes to incorporate the evolving information. This “movement” occurs through either the refinement of the average position, or the application of proper motion to the time-history of detection positions. Additionally, it is possible for the time-histories to show that a single object should be split into two objects (or even that two objects should be merged as one).

Although the IPP will be producing a data record each time a celestial object is detected, it does not provide a record for the object itself. Thus, the celestial object record within the ODM must be updated based on the complete set of detections, which are not expected to be maintained by, nor consulted during, the IPP processing.

### **2.3 Expected PS1 Publication Counts and Rates**

The small record size and simple structure of the ODM’s published products are countered by large record counts and high processing rates. For this discussion we reference only order-of-magnitude numbers and rates, employing a single digit of precision. Additional precision on the numbers will not change the underlying nature of the ODM data management problem.

Table 1 enumerates the number of published entities expected to be produced in a single year of operation for PS1. Note that these rates are per day, not per observational night. PS1 is expected to process data every day of the year (within the required availability), even if Pan-STARRS is unable to capture images every night of the year.

A bit of context is required to clarify the organization of these records. For the cumulative detections, there will be a single detection record for each color filter on each celestial object (within a single year). On the other hand, the P2 detections will occur against a smaller set of brighter celestial objects (on the order of one billion objects). These P2 detections will occur several times in each color filter. The 5 $\sigma$  delta detections will usually be associated with a known celestial object, but not always. The 3 $\sigma$  delta detections will not be directly associated with celestial objects within the ODM, since they will not be accessible individually.

The “Publish Action” in Table 1 indicates how the information is required to be integrated with the existing published products. In the case of all IPP-produced entities, the ODM need only add them to the existing information. On the other hand, the ODM-derived celestial object records must be updated when associated detections are published. Naturally, it is necessary to add a new celestial object record the first time an object is detected. After one year, most of the possible celestial objects will have been seen, and a corresponding record will have been created within the ODM. Therefore, the count of celestial objects will remain essentially constant after the first year. However, the counts of the other record types will continue to grow at the same rates.<sup>2</sup>

---

<sup>2</sup> It is possible that new cumulative detections will replace existing cumulative detections in successive years. This decision has the potential to trade-off initial development complexity for reduced long-term storage requirements (in a long-running PS1 system).

## 2.4 ODM Access Patterns

The published ODM products will be accessed in two distinct modes: externally by end-user astronomers<sup>3</sup> and internally by the ODM itself. A complete set of end-user access patterns is inherently unforeseeable, since the nature of research is to ask questions that have not been asked before. The internal access patterns are more predictable, since these arise from the anticipated processing required to publish and organize the products themselves, and will be discussed in that context in the next section.

An important feature of the Pan-STARRS survey products is the time-history made possible by repeated observation of the sky. Although the end-user access patterns are difficult to predict, it can be reasonably assumed that access to this time-history will be important. Further, it is likely that the time-history will be accessed on a per-object basis, rather than by location on the sky. That is, end-users are likely to be interested in the time-history of detections on a specific celestial object rather than in the time-history of a general area of the sky.

All of the ODM products have an explicit spatial nature; that is, they are all at some location on the sky. This implies a spatial access pattern, in addition to the temporal access patterns for detections.<sup>4</sup> These two access patterns can be combined either directly or indirectly to construct most—possibly all—significant end-user access needs. We provide the following list as a summary of potential access patterns:

1. Time-History Access: Find all detections for a specified celestial object;
2. Simple Temporal Access: Find frames or detections that occurred within a timeframe;
3. Simple Spatial Access: Find frames, detections, or objects that lie within a region of the sky;
4. Simple Spatial-Temporal Access: Find detections that occurred within a timeframe for all objects that lie within a region of the sky;
5. Complex Spatial-Temporal Access: Find frame metadata records or detections that occurred within a timeframe and lie within a region of the sky; and
6. Standard Relational Access: Find frames, detections, or objects that match other attributes, e.g., magnitudes, shapes, color, etc.

We note that it may be possible to rearrange some of the category 5 patterns to fall into category 4 by use of spatial access to the frames that contain the detections. Naturally, the standard relational patterns are expected to be applied in combination with all other patterns.

It is interesting to note that the published IPP data products are provided to the ODM previously organized for certain temporal access. That is, the ODM will receive detection records associated with frames, which have location and time attributes, and the frames generally arrive in chronological order. These explicit spatial-temporal relationships will be available for access, regardless of how the published data are reorganized within the ODM. However, the time-ordered data coming from the IPP will not arrive organized in a manner that readily supports time-history access for individual celestial objects and their associated detections. Reorganizing the products to support both time-history and spatial access patterns is potentially complex, and is the subject of the following section.

## 3 ODM Ingest Processing Summary

Readers familiar with traditional data management design and development are well aware of the challenges imposed by large record volume. However, the Pan-STARRS ODM also poses a significant challenge in the

---

<sup>3</sup> End-users will access the ODM through the Web-Based Interface (WBI) and Data Retrieval Layer (DRL) components of the PSPS. There will not be any direct end-user access to the ODM component itself.

<sup>4</sup> Note that use of the term “spatial” does not imply a specific solution (e.g., a vendor product), but rather that the search is based on multi-dimensional distance calculations instead of the matching of simple keys. There are several promising solutions to this problem available within the commercial and academic arenas.

processing required to ingest new data records for insertion into appropriate organizations within the data stores. Such insertion processing must keep pace with the data collection and the IPP generation. Of course, these data cannot be write-only records, as the goal of Pan-STARRS is to provide access to scientists for their continuing research. The paragraphs that follow present a discussion of the ingestion needs to be met by the ODM.

Published products will arrive from the IPP organized according to the observational process. That is, the IPP will deliver detections associated with their frames in the chronological order that the frames were taken. The ODM must maintain and update the derived products (i.e., the celestial object records) as new data are published. Additionally, the ODM should organize this information to enable as many access patterns as possible, and to facilitate the most likely patterns. Since keeping pace with the IPP publication rates requires that these internal processes be as efficient as possible, the intent is to optimize the access patterns that are used internally by the ODM to perform its own maintenance and organization. This approach is reinforced by the difficulty in predicting the most likely end-user access patterns, as discussed in the previous section.

For purposes of describing the ODM challenges, we discuss both a physical organization and a logical organization. The physical organization refers to the actual location of data products within the hardware and software of the ODM. The logical organization refers to the techniques and procedures that allow the ODM to know where a product is within the physical organization. The important distinction is that a physical organization implies the existence of limits or boundaries. Additionally, there can be only one physical organization scheme for a product, but there can be multiple logical organizations. For example, the physical organization might involve a number of servers and disks, each with a product capacity limit. This paper does not propose or reject any particular physical organization, but does identify where the existence of boundaries or limits should be considered during architectural design.

What follows is a description of the general data processing required within the ODM to maintain the derived celestial object properties and to organize the IPP data to support this maintenance. The organization process involves the following activities:

- Frame metadata loading,
- Detection-to-object correlation,
- Detection loading,
- Celestial object creation, and
- Celestial object maintenance.

After describing these organization activities, we discuss how the resulting internal access patterns relate to possible end-user access patterns.

### **3.1 Frame Metadata Loading**

The record data for new frames are added to the existing published records within the ODM. The expected rate is easily handled through traditional techniques, which will allow ready access via both spatial and temporal attributes. This is not likely to be affected by the physical organization, since the total size should fit well within any likely physical boundary or limit. Once a frame record has been loaded, it will not be updated or changed.

### **3.2 Detection-to-Object Correlation**

Upon arrival at the ODM, each new detection record is associated with a (unique) known celestial object, if possible. In the likely case, this correlation is done through a spatial search of existing celestial object records.

If the IPP is able to provide a unique identifier to a known celestial object, then this correlation may be done through a more direct key-based search. In this case, though, the ODM would need to manage the issues that arise when the

IPP's internal object records are out of synch with the current ODM object records.<sup>5</sup> A frequent feedback from the ODM to the IPP is not permitted, so the need for some amount of spatial detection-to-object correlation will always remain necessary within the ODM.

### **3.3 Detection Loading**

The various detection records for each frame are added to the existing published detection records. The anticipated ingestion rate and total record counts are very high by current technology standards. The records arrive from the IPP physically organized by frame, which provides either a spatial or a temporal organization. To support both types of access—expected for end-user access, and required for internal ODM processing—additional logical organization must be created.

For example, if the detections are physically organized with a spatial arrangement (i.e., put all detections for a region in the same physical boundary), then an additional logical organization must be created which identifies how to access the detections using temporal attributes. For example, the additional logical organization might imply a process to look at every physical storage location for detections that occurred in a given timeframe.

This said, for purposes of maintaining the celestial object properties, it is important to have an organization that provides efficient access to all of the detection records associated with a given celestial object. This celestial object maintenance process is detailed in the following sections.

The good news about the management of these vast numbers of entities is that once detections are loaded (after the correlation process), those records are not likely to be updated or changed. A rare exception would be to change the association with a celestial object, perhaps because additional observations have resulted in a split or merge of celestial objects. Another exception would be the removal of a set of detection records if, for any reason, it was determined that the IPP needed to reprocess those detections and replace them.

### **3.4 Celestial Object Creation**

A new celestial object record must be created for any incoming P2 or cumulative detection that cannot be reliably correlated with an existing object. The frequency of creation of new objects is expected to fall off as the sky is repeatedly observed, countered by the addition of new observable sky as the Earth moves around the Sun. After the first year, the creation of new celestial object records is anticipated to drop to a very low rate. Celestial objects created previously will need to be updated, as described in the following paragraph.

### **3.5 Celestial Object Maintenance**

Ideally, the celestial object properties should be revised every time the object is observed (i.e., a new detection record is associated with the object). For most objects, the first few detection records will provide the most significant changes to the object properties, and most objects will reach a steady state after several successive observations in all filters. Rarely, it may be necessary to revise object properties when detections are removed for reprocessing.

As discussed in Section 2, the various categories of derived attributes may be generated on different schedules, as appropriate to the input data required to calculate the attribute, and the universal nature of the attribute itself. In conjunction with the scheduling of attributes, celestial object maintenance occurs in two distinct modes:

---

<sup>5</sup> Note that any amount of detection-to-object correlation within the IPP may additionally complicate the ODM publication process. The intensive, but simple, spatial correlation process becomes a challenging data cleansing problem, since the IPP may publish correlations based on sky positions that do not match the current state of the ODM's published celestial objects, whose sky positions are based on the evolving time-history.

- **Visible Sky Maintenance:** Roughly half of the sky will be visible on any particular night, and P2 detections could be generated for any object in the visible sky. It is anticipated that only (roughly) 10% of the possible objects will be visible at the magnitudes expected for P2 detections.
- **Cumulative Sky Maintenance:** Each day, a 1° wide slice of sky directly opposite the visible sky will be processed for cumulative detections. The objects in this slice will receive a new detection record for each filter.

Regardless of physical organization, maintaining the object properties will be a relatively intensive process. To maintain a celestial object, the full time-history of detections must be gathered and then the latest aggregate properties must be calculated. The existing celestial object record must then be updated, which is always a more expensive action than simply adding a new record.

To mitigate some of the expense, it is possible to defer this processing and do it in batches. By waiting for the arrival of several detections of the object before revising the object properties, we can perform the maintenance in a single step. However, deferred maintenance can only occur when P2 frames overlap within an acceptable deferral period. The nominal daily update rate identified in Table 1 presumes that there is minimal overlap of P2 frames, but that the cumulative sky maintenance is deferred until all filters are loaded.<sup>6</sup>

The physical organization of the celestial object records must be carefully considered. Since the calculated sky position of a celestial object will be refined by further detections, any physical organization that is spatially-based must be able to handle the relocation of object records. None of the celestial object properties lend themselves well for use in physical organization schemes, since none are immutable (except for a unique identifier).

### 3.6 ODM Internal Access Patterns

The above paragraphs identified the processing required for the ODM to organize and publish the IPP products. To support this internal processing and meet the anticipated ingestion rates, the following access patterns must be well-supported and efficient:

- Simple spatial access to the celestial objects records, i.e., find objects near a sky position; and
- Time-history access to the detections associated with a celestial object; i.e., find all detections associated with a specific object.

All end-user access patterns identified in Section 2 are enabled indirectly through the addition of standard relational access support (e.g., find all detections associated with a specific frame) and spatial access to the frames themselves. Although the complex spatial-temporal end-user access patterns may not be as efficient as those patterns that are directly optimized for ODM ingest processing, they certainly are achievable.

## 4 Unique Characteristics of the PSPS ODM

The above paragraphs present an introduction to the ODM issues that must be addressed in the PS1 operational prototype. In this section, we highlight the characteristics that make the ODM different from extant systems—especially extant astronomy survey systems—that might initially appear comparable. Some of these characteristics increase the challenges posed by the ODM, and some simplify the challenges. All of these characteristics are important factors when discriminating possible solutions for the ODM design. These unique characteristics are:

**Fast:** Near real-time ingest rates,

**Simple:** Simple entities and relationships,

---

<sup>6</sup> That is, the object update rate is the sum of the P2 detection rate, the cumulative detection rate divided by the number of filters, and the non-noise  $5\sigma$  Delta detection rate. The values in Table 1 are rounded to a single digit of precision.

**Spatial:** Multi-dimensional distance calculations,

**Large:** Massive number of managed entities, and

**Lenient:** Academic and prototype features.

It should be noted that there are existing commercial, government, and academic data management systems that possess several of these characteristics. However, the PS1 ODM appears to be the first system planned to “enjoy” all of them, especially to the expected degree. Each of these characteristics will be detailed below.

## 4.1 Fast

The ODM needs to operate at near real-time ingest rates. In other words, the ODM cannot fall too far behind the IPP’s generation of published products. This means, on average, that the ODM must add 200 million various detection records and update 100 million celestial object records per day.

These rates are comparable to those seen in telecommunications call-detail data stores. However, the ODM must perform some form of a correlation search to associate each detection record with the appropriate celestial object, while the telecommunications systems simply append new records.<sup>7</sup> Each update of a celestial object requires a full time-history search of all detections associated with the object. The ODM’s ingestion rates are unique among published astronomy archives, which have not been required to concurrently perform collection, data processing, and data management for publication purposes.

## 4.2 Simple

The published entities themselves are relatively simple, with few attributes and uncomplicated relationships to other entities (once the correlations have been performed). The possible exception to this is the  $3\sigma$  Delta Detection sets, which may ultimately require some form of “unpacking” to access their internal detection records. The number of these sets is relatively small compared with object and detection counts. Additionally, the relationships to the sets themselves are very simple: a one-to-one relationship exists between a  $P4\Delta$  frame record and a  $3\sigma$  Delta Detection set.

Another aspect of this characteristic is the fact that detection records can generally be considered immutable once they have been correlated to their appropriate celestial object. This can allow for significant simplifications in the physical and logical organization of these records. Although the ingestion rate of these records is unusual for astronomy projects, ODM data records are a bit simpler—and hence smaller—than those stored in many extant astronomy archives.

## 4.3 Spatial

All of the ODM published products include an explicit position on the sky. This requires robust support for multi-dimensional distance and area evaluations on the surface of a sphere. Such evaluations cannot be supported by traditional relational tools, which are generally key-based and perform their search through equality or range comparisons on a single attribute. However, several solutions (commercial and custom-developed) exist that address this need. The key factor is that all of these solutions require significantly more processing capability than a simple relational access.

## 4.4 Large

The ODM will manage a massive number of individual entities. The record counts for a single year of PS1 operation approach 100 billion. Regardless of the physical and logical organization, this record count is certainly at the cutting edge of data management technology and far exceeds extant astronomy survey projects.

---

<sup>7</sup> This correlation search will usually include a spatial search of the known celestial objects within the ODM.

## 4.5 Lenient

Mitigating the impact of some of the ODM's stressing characteristics is the fact that the PS1 ODM is an operational prototype of an academic research tool. It will be neither a high-availability commercial business-enabling system nor a high-volume/high-performance intelligence analysis system.

The operational prototype nature of PS1 allows for certain simplifications. For example, the PS1 IPP is expected to maintain all images and intermediate products for the duration of the PS1 operational life. This prototype flexibility allows the ODM to avoid expensive reliability and backup solutions by simply asking the IPP to re-publish lost data.

The academic/research nature of the full Pan-STARRS system, as well as the PS1 operational prototype itself, allows for significant cost savings in the area of availability. The availability requirements for the ODM are significantly lower than would be expected in a commercial or public-sector application. In fact, as long as the ingest processing is able to continue, time-consuming repair actions on large portions of the physical organization can be tolerated during operation.

## 5 Distribution of Resources

In an ideal world, a computing problem could be solved using a single set of storage, processing, and application resources. Unfortunately, many problems exceed the physical or logical boundaries of such components. When that occurs, the problem must be partitioned in some manner to fit within those boundaries. When data volume exceeds the size of a single storage device, records must be split among files on different devices, or multiple devices must be aggregated to appear as a single, larger storage device. When computational requirements exceed the capacity of a single processor, processing must be split among multiple processors to "divide and conquer" the problem. Applications, in turn, are partitioned to allow their use of processing and storage to fit within the physical and logical boundaries of the other resources.

The decision to distribute a resource is often a simple choice based on the problem being solved and the technologies available and/or affordable. Any data management system of significant size will require distribution of data among storage resources. It is also likely to require distribution of data processing among processors, and may require distribution among applications. The complexity of managing and using the distributed resources will vary with the degree of distribution and the level of support for resource distribution that is provided by the underlying technologies.

Scalability issues are often behind decisions to distribute resources. In addition to exceeding storage boundaries, large data management problems are often constrained by other boundaries, such as the address limits of an application or the memory within a processing node. Partitioning this type of problem across boundaries takes two forms:

- **Scaling Up:** Scaling up increases the resources available within a single component. Examples include adding CPUs or memory to a processing node, or adding disks to a storage array.
- **Scaling Out:** Scaling out adds extra components to provide additional resources. Examples include adding processing nodes to a compute cluster, or standing up additional web servers in an e-commerce site.

Each scaling method is best suited to certain types of problems, and neither is without limitations. At some point, a node runs out of room for memory or processors. Similarly, a cluster will run out of interconnect bandwidth at some number of nodes.

When system developers choose to distribute a problem across resources, then they must decide how the information being processed will be partitioned across the boundaries of those resources. This decision requires choosing a physical organization for that information. A good organization will optimize the use of available resources; a bad organization results in consumption of additional resources. In determining the organization, developers desire to minimize four factors:

- **Management Communication:** Reduce the control information that must be exchanged among resources to synchronize their activities,
- **Data Movement:** Avoid moving data from one resource to another,
- **Interference:** Avoid resource contention where processes have to wait for others to finish an activity before they can continue, and
- **Load Imbalance:** Avoid hot spots or idle resources.

Minimizing these organizational factors in the context of a real problem, including factors such as allowable complexity and budget, requires the development of an architectural strategy. Architectural strategies are enumerated in the following section.

## 6 Architectural Strategies

Architectural strategies for modern data management systems can be described in terms of resource distribution and pooling choices. The level of *distribution* of a resource indicates a decision to partition a problem into pieces, either to address a resource limitation or to make the problem more tractable. The level of *pooling*, or virtualization, of a resource indicates a decision to combine results from, to balance the usage of, or to streamline the management of that distributed resource. Resource pooling often entails increases in technological complexity, so the decisions should be made consciously, and in proportional response to the problem being solved.

As discussed in the previous section, the decision to distribute resources will often follow directly from the nature of the problem and the technologies available to address that problem. The decision to pool those distributed resources, however, is less straightforward. In the subsequent paragraphs, we introduce three orthogonal classes of resource pooling, describing the benefits and disadvantages afforded by various pooling decisions in each class. We then use these three classes to describe the set of possible architectural strategies.

### 6.1 Resource Distribution and Pooling

Data management systems require distribution and pooling decisions for three resource classes:

- **Application Pooling:** Software subsystems range from independent applications that require end-user integration of input and results, to clustered operating system and distributed application options, including federated, independent databases.
- **Processor Pooling:** Processor subsystems range from independent (share-nothing) processing nodes, through clustered computer options, to large symmetric multiprocessor (SMP) options.
- **Storage Pooling:** Storage subsystems range from internal direct-attached disks, through external disk subsystems, to various networked storage options, such as network attached storage (NAS) or storage area networks (SAN).

These three resource pooling classes are discussed independently in the subsequent sections. The benefits and consequences of pooling decisions within each class follow.

#### 6.1.1 Application Pooling

Application *distribution* divides a problem among multiple instances of the same application.<sup>8</sup> Application distribution can be used to avoid scalability issues with a single instance, or to improve performance for geographically dispersed users. Application *pooling* combines multiple instances of the same application to present unified results to the end-user or software clients. Pooling may be used to simplify users' requests for data, or for load-balancing between instances when user or system workloads vary. Pooling application instances across multiple sets of

---

<sup>8</sup> This is different from application pipelining where a problem is partitioned into different (sequential) stages that are handled by separate applications.

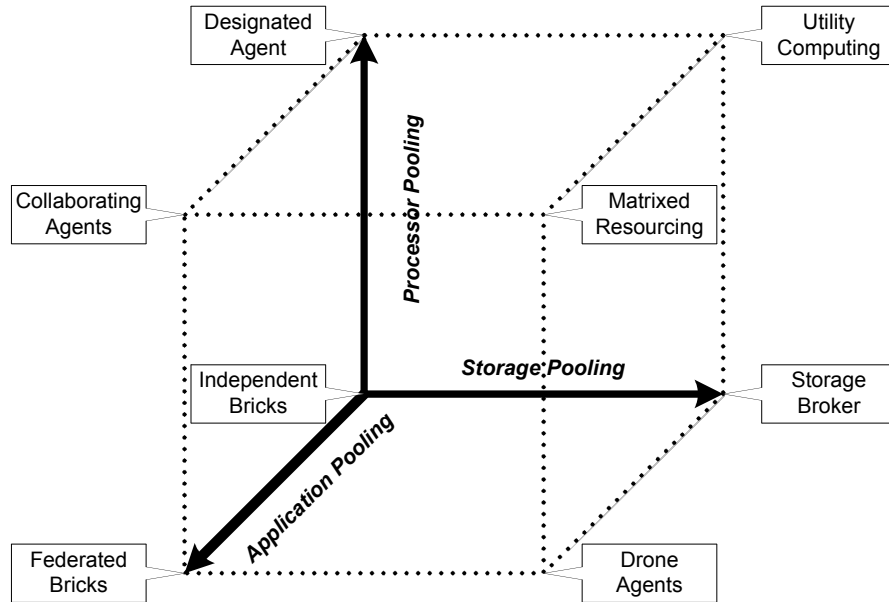


Figure 2. Decision Space for Architectural Strategies

hardware may increase availability through fail-over, or allow multiple small boxes to solve a large problem at less cost than a single large box.

On the down side, pooling adds application complexity. If the application instances are completely independent, then users must know which one contains data of interest. When pooled, a process must orchestrate the workload through the separate instances so they appear as a concerted system. If data are visible only within an instance, then load balancing across instances depends upon data access patterns matching the data distribution scheme. Data can be made visible across instances to improve load balancing, but some process must then coordinate reads and writes to ensure data consistency.

At one pooling extreme, a data management system might be comprised of a number of independent data managers, each responding directly to requests for the data they contain. The other extreme might be a large corporate data warehouse or e-commerce system with many concurrent users connecting to a single application.

### 6.1.2 Processor Distribution and Pooling

Processor *distribution* divides a problem among multiple CPUs and memory. This provides computational throughput by splitting the work among multiple inexpensive processors rather than using a single high-performance (and high priced) one. Distribution can avoid scalability issues with a single processor system (e.g., a mainframe) and enable load balancing between processors. Processor *pooling* coordinates the work of multiple CPUs and memory so that their work stays synchronized. Pooling processors across multiple sets of hardware may increase availability, through fail-over, or support additional load balancing. Recently introduced technologies, such as virtual machines and blade servers, allow dynamic allocation of resources to processing pools.

The primary concerns with processor pooling are: (i) the degree of parallelism achievable, given the problem being solved, and (ii) the ability of the processing scheduler to allocate work efficiently. In systems with independent processors and no scheduler, it is likely that some processors would be idle while others were overloaded. The problem is compounded if the data under management were visible only to the local processors. On the other end, a single multi-processor computer can have internal complexity with memory allocation or inter-process communications that limit the number of processors that can be installed effectively.

As example, one pooling extreme might be a set of PCs independently running a Monte Carlo simulation with different parameters. The other extreme might be a high-CPU count system used for computer generated animation.

### 6.1.3 Storage Distribution and Pooling

Storage *distribution* divides data among storage devices. This allows management of large volumes of data. Storage *pooling* aggregates storage devices to make them accessible to the applications and processors. Storage pooling improves throughput and load balancing with multiple I/O controllers working in parallel, and by striping data across multiple storage devices. Storage manager software supports the creation of logical drives larger than a single device, which simplifies management as well as supporting growth (i.e., provisioning) over time. These software managers also help protect against device failures by using redundancy techniques. Allowing access to pooled storage by multiple processors and applications through some form of network connection facilitates load balancing of data requests and permits device consolidation.

With independent storage devices, applications access each physical device directly. However, independent storage devices limit file sizes. A storage manager is desirable to mitigate this limitation and can protect against device failure. Using local storage, even with load balancing, constrains data organization to match access patterns. Mismatched growth patterns may require extra storage in some local stores, while others have unused capacity.

A management process and some form of network connectivity are required to make storage visible to multiple processing subsystems, and to manage their read/write requests. This connectivity, in turn, requires more complex storage management software.

A workgroup network, with no file server, is an example of an architecture with no storage pooling. Files are stored locally on each computer, requiring users to access data files from separate computer file systems. At the other extreme one might find a large corporate data center with multiple servers sharing a large, centrally managed pool of storage.

## 6.2 Possible Strategies

Any complete architectural solution must consider the interactions between the three resource pools in the context of the problem being solved. To discuss the range of architectural strategies, though, we can consider these pooling decisions to be independent.

If we consider each pooling decision as an independent axis, the exhaustive set of possible architectural strategies can be described using the cube in Figure 2. The vertices of the cube indicate the extreme combinations of all-or-nothing resource pooling. Each vertex is a viable architectural strategy, and problems exist that can best be solved by each of those strategies. The strategies are enumerated below:

- **Independent Bricks:** “Fat bricks” without a federated data manager. The user knows which machine to access and then integrates results from multiple machines, if necessary.
- **Federated Bricks:** “Fat bricks” with a federated data manager. The user submits a single query and the results will be integrated before returning. Example: Sloan Digital Sky Survey.
- **Designated Agent:** A compute cluster with direct attached disks and individual applications, or a large multi-user Symmetric Multi-Processor (SMP) system. Example: Hosted J2EE web applications
- **Storage Broker:** Independent processors without a federated data manager connected to a SAN or networked file system. Example: Storage Resource Broker ([www.sdsc.edu/srb](http://www.sdsc.edu/srb)).
- **Collaborating Agents:** A compute cluster with direct attached disks, but using a single entry-point application. Example: SETI@Home.
- **Drone Agents:** Independent processors connected to a SAN (or analog), all working on the same problem. Example: Human Genome Project.

- **Utility Computing:** A “grid” data center: Virtualized storage and processors, but multiple applications serving various needs. Example: Corporate Management Information Systems (MIS); Smallpox Research Grid Project.
- **Matrixed Resourcing:** A compute cluster with federated data management and virtualized storage, working on a single problem. Example: AT&T Call Detail Warehouse.

Each of these strategies is a viable approach for certain problems. The names are notional, and are intended to evoke the underlying concept of each strategy. The next section examines candidate technologies available to implement these strategies.

## 7 Available Technologies

Architectural strategies are implemented by assembling components into a data management system. The principal components of a data management system can be thought of as layers. The layers defined in this section are generic, and no specific technology or implementation is assumed.<sup>9</sup> However, decisions of budget allocation and vendor selection depend on consideration of candidate components/layers. As a result, we provide a brief introduction to current technologies available to implement an architectural strategy in each of the following layers:

- Application Layer
- Processing Layer
- Storage Layer

Although all layers must be represented in a data management system, individual components might be implemented in a number of ways, depending on the architectural strategy selected. The following paragraphs provide a definition of each layer and list the range of technology products that exist for that layer.

### 7.1 Application Layer

The application layer is responsible for efficiently storing and organizing records, retrieving them and managing their updates. This layer runs on the computing resources in the processing layer. As multiple processes are likely to be accessing data simultaneously, the data manager must ensure the integrity of concurrent read, insert, update, and delete operations. While one implementation of this layer might be a Database Management System (DBMS), this layer could also be implemented as a file system with scripts to search its files, or a mix of the two (or something else entirely). Nothing presumes this layer is a single process or instance. It could be a group of separate data managers working independently, or coordinated in some manner.

The application interconnect is used to move control information and data among instances in the application layer. Federated or clustered databases use their interconnect to de-conflict reads and writes and to move results sets across instances. A single instance database would not need an interconnect, since that function is handled by its internal scheduler.

Relevant technologies in the application layer are file systems and Database Management Systems. File systems handle shared access to files. A DBMS handles shared access to records stored within files or on “raw” devices assigned to the DBMS. Types of file system include single computer systems, network file systems supporting shared access across networks, clustered file systems supporting applications doing concurrent read and write, and hierarchical file systems that extend functionality to non-disk media, such as tape or optical, to reduce the cost of retaining infrequently accessed data. Both files systems and DBMSs support shared access to data. With a file system, files are generally shared for reading, but locked when a user or process is writing or updating the file. With a DBMS, locking occurs at the record level.

---

<sup>9</sup> We note here that our Pan-STARRS efforts are in the preliminary design phase, and as such this paper does not address the technology “solution space,” rather only alternate architectural candidates currently under investigation.

## 7.2 Processing Layer

The processing layer of a data management system contains the computing power (CPU and memory) and operating system that the data management processing requires. This layer is responsible for supporting the data manager's processing, and for ensuring the stability of active processes. In this context, there is no presumption regarding the number of processors, how they are connected to memory, nor how these processors interact (i.e., communicate) with the storage layer.

The processor interconnect is used to move control information among nodes in the processing layer. For a single node, its interconnect might be the node's internal cards and cabling. For example, an SMP computer might use its backplane to pass control information among its processors and memory to implement the processor interconnect.

The processing layer's components are CPUs and memory within some number of processing nodes. These processing nodes could range from single CPU machines through small Symmetric Multi-Processor (SMP) computers to enterprise-class SMP systems or mainframes. Virtual machines, a software technology that allows multiple operating system images to run on a single node, are also candidate implementations for this layer. A recently introduced technology called Blade computing allows hardware clustering to dynamically allocate processors and memory to virtual machines. Processor interconnect products include cabling, switches, hubs, and routers. A number of protocols exist for networking processors. These include Ethernet, Gigabit Ethernet, and Infiniband, among others.

## 7.3 Storage Layer

The storage layer of a data management system contains the storage devices and their controllers. It consists of a combination of direct-access and sequential-access storage devices. This layer might include either internal or external storage devices.

The storage interconnect is used to move data between the processing layer and the storage layer. The SMP computer, identified in a previous example, could use the same backplane as the processor interconnect to access locally installed disk drives, or it might use network interface cards and cabling to external storage.

Technologies in the storage layer include storage devices, controllers, and interface devices. Storage devices of interest include magnetic disk and tape. Disk and tape drives come in a variety of sizes and support various serial and parallel interfaces. For both technologies, access performance declines as the capacity goes up. Controller and interface technologies include direct disk or RAID for internal storage and network interface cards or host bus adapters for external storage. The storage devices themselves can be installed internal to processing nodes, or in external storage arrays and tape libraries. Storage interconnect products include the same components and protocols that are available for the processor interconnect. This interconnect can also use block mode protocols, such as iSCSI and Fibre Channel, which may provide improved throughput.

## 8 ODM Characteristic

Support of ODM functions will require efficient spatial and time-history access to published products, both of which are expected to be important end-user access patterns as well.

Several characteristics are highlighted that distinguish the ODM from existing systems. All of these characteristics are important factors when evaluating architectures for the ODM. Taken individually, the stressing characteristics identified above (dynamic, fast, spatial and large) are not uncommon in the data management domain. The combination of all of these characteristics, however, poses a particularly challenging problem. The interaction of certain characteristics leads to obvious tension (e.g., fast and large, dynamic and fast). Fortunately, the mitigating characteristics (simple and lenient) can reduce the problem, and potentially lower the cost of possible solutions.

The combination of the spatial and large characteristics likely implies a physical organization that is spatial in nature, but care must be taken to manage the complexity of the logical organization schemes that will be required to

**Table 2. Optimization Goals for the ODM**

<b>Activity</b>	<b>Optimization Goal</b>
Ingest	Sustain constant ingest rate as total record count increases.
Ingest	Scale storage linearly with the total record count.
Ingest	Support increased ingest rate through linear addition of processing hardware.
Query	Support increased user query load through linear addition of processing hardware.
Ingest/Query	Support the addition of storage hardware without disruption.
Ingest/Query	Support the addition of processing hardware without significant disruption.
Ingest/Query	Support variable ingest and query load balances without introducing delays.
Query	Constant query performance on object records.
Query	Linear query performance on detection records proportional to the number of records retrieved.
Query	Support query patterns for objects and detections as they are observed during operations.
Query	Allow tuning of storage access based upon observed end-user access patterns.

support any physical organization. For example, the difficulty of managing overlapping areas or duplicated records must be properly considered during design.

Taken individually, the stressing characteristics (fast, spatial and large) are not uncommon in the data management domain. The combination of all of these characteristics, however, poses a particularly challenging problem for the ODM. The interaction of certain characteristics leads to obvious tension, e.g., fast and large, dynamic and fast. Fortunately, the mitigating characteristics (simple and lenient) can reduce the problem, and potentially lower the cost of possible solutions.

## **8.1 ODM Desired Behavior**

Before evaluating architectural alternatives, it is important to consider the desired behavior of the system. The following paragraphs address this in the context of: (i) the ideal features that would enable the ODM to perform its job effectively, (ii) the technical risks that we foresee (and which must be mitigated), and (iii) the ODM characteristics to be optimized. These three qualities can be described in terms of goals:

- **Feature Goals:** Data access is the principal function of the ODM. It must be able to support access efficiently, which means being able to organize data for efficient ingest and query. The ODM should achieve optimum use of its resources, especially processing and storage. The configuration must be manageable without extraordinary efforts by Pan-STARRS administrators. In order to reduce initial costs, ODM storage must be able to grow incrementally as data are collected over time. The ODM implementation should minimize the demands it makes on infrastructure, such as space, power, and cooling. Finally, as with all system development efforts, the authors are especially cognizant of the project's budgetary constraints, and desire to develop and deploy the ODM at a reasonable price consistent with getting its job done.
- **Risk Mitigation Goals:** Some aspects of the ODM pose technical risks to be mitigated. First, the number of records to be managed is extremely large. Second, the resulting volume of storage may strain the data management and processing layers' capabilities to access that storage. Last, given the size and potential complexity of this system the authors desire to assure a level of availability consistent with implementing the ODM at a reasonable cost. Without such assurance, a significant amount of time and money might be expended during the ODM's operational period.
- **Optimization Goals:** Given alternatives that have the desirable features, as well as mitigate the risks, then the qualities to be optimized must be considered. These qualities are enumerated in Table 2, both for ingest processing and end-user query activities.

It is noted that these are target goals for ODM behavior, not system requirements. These are the qualities desired to be achieved when considering architectural alternatives, and they must be traded with cost and complexity factors.

## 9 ODM Architectural Implications

The characteristics and desired behavior of the ODM inform architectural decisions. The first of these decisions relates to the distribution of resources. The volume of storage required will certainly exceed the physical limits of any available storage devices, implying a distributed storage solution. At the same time, the required high processing rates indicate a distributed processing solution. An architectural strategy must be chosen to handle these distributed resources, and component features must be considered.

Architectural components of a system are selected to support the chosen pooling strategy, optimize the desired behavior, and minimize cost and complexity. The stressing and mitigating characteristics of the ODM are reiterated in Table 3, along with their implications for the component layers. The following paragraphs present these implications in the context of pooling strategies for the ODM.

### 9.1 ODM Resource Pooling Strategies

Pooling decisions can be made independently for each of the data management resources. The pooling implications for each resource are discussed in the context of the ODM characteristics and desired behavior.

#### 9.1.1 *Application Pooling Implications*

The data management application pool should appear to be a single data manager. Query splitting and results aggregation should be invisible to the end-user. Similarly, the detection load and object maintenance processes should require neither access to, nor information about, the internal organization of data handled by the data manager. This isolation allows the data management application to be restructured or reorganized without outside impact.<sup>7</sup>

Operations should be highly parallelized within the data management application. Ingest processing and object maintenance each require a significant amount of storage interaction. To increase their performance, the data manager must be able to partition, and isolate, ingest and update operations for data representing different parts of the sky. This allows processes to run in parallel without contending for the same resources. As a result, the data management application should organize data spatially to minimize the I/O required to support object updates.

An optimal configuration includes data that are globally visible to the management application instances. In this case, each instance could process data in any region to support load balancing. This configuration would also reduce complexity by reducing the required number of data management instances, thus minimizing coordination among instances. However, this configuration may conflict with the partitioning and isolation needed to support parallelization. If data are to be partitioned among independent data management instances, then an optimal design must avoid duplicating or replicating data across instances. Replication adds significant complexity and workload in order to keep copies consistent, creates opportunity to introduce errors, and increases the volume of data that must be managed.

An attractive application pooling approach would include a single data manager instance, or a small number of instances clustered together. Data would be globally visible to the application or cluster, so that any instance can service any request. The data manager would have internal capabilities to support partitioning, parallelization, and isolation with respect to processor and storage utilization. The data manager would be able to organize data to minimize I/O operations.

#### 9.1.2 *Processor Pooling Implications*

A goal in pooling processors is to spread the work-load across all processors as evenly as possible, in order to reduce the number of processors in the system. With respect to ingest and update—the ODM's most challenging functions—multiple, parallel processing threads are essential to achieve the necessary throughput.

The allocation of work to processors should be invisible within the processing pool. Neither the application pool, nor the storage pool, is required to be aware of processor assignments. This transparency facilitates dynamic allocation of processors and keeps the layers isolated from each other.

It is reasonable to assume that some number of processors will be aggregated into nodes, which in turn are aggregated to achieve the processing power required. This approach optimizes inter-processor communications through faster internal communications paths without placing too many processors in a node and thereby exceeding the node's effective capacity.

For the configuration to be efficient, the processing nodes must be sized to minimize workload caused by cross-node communications and minimize data movement among nodes. Ideally, each processor would be able to see data globally. Alternatively, if processors in the pool could only see data for a portion of the sky, then those processors would be idle when data in that region was not being processed. In this sub-optimal case, data must be moved between nodes to put those idle processors to work. Such manipulation increases the bandwidth requirement on the processing node interconnect, and could slow down the overall system.

A desirable processor pooling approach includes a number of processors in multiple nodes that are clustered together. Data would be globally accessible for all processors, so any processor would have access data without going through another processing node.

**Table 3. Implications on ODM Architectural Pools**

<b>Characteristic</b>	<b>Application Pool</b>	<b>Processor Pool</b>	<b>Storage Pool</b>
Fast	<ul style="list-style-type: none"> <li>• Ingest at required rates</li> <li>• Parallel threading and isolation</li> <li>• Prevent management overhead from slowing ingest</li> </ul>	<ul style="list-style-type: none"> <li>• Allocate workload across all processing resources to meet processing complexity needs with minimum components</li> </ul>	<ul style="list-style-type: none"> <li>• Optimize organization to minimize I/O operations used to correlate detections to objects and to retrieve time-history of detections for an object</li> </ul>
Simple	<ul style="list-style-type: none"> <li>• Data structures amenable to relational DBMS</li> <li>• Take advantage of read-only status to reduce system overhead</li> <li>• Store small records efficiently</li> </ul>		<ul style="list-style-type: none"> <li>• Efficiently support storage/retrieval of small records</li> <li>• Small records size suggests that high bandwidth interconnect may not be needed throughout</li> </ul>
Spatial	<ul style="list-style-type: none"> <li>• Support spatial correlation and query</li> </ul>	<ul style="list-style-type: none"> <li>• Parallel threading required</li> </ul>	
Large	<ul style="list-style-type: none"> <li>• Manage number of records</li> <li>• Capable of partitioning records in some manner to support scale out rather than scale up</li> </ul>	<ul style="list-style-type: none"> <li>• Enough sophistication to address the volume of storage under management</li> <li>• Storage interconnect able to scale out to the full volume of data without bottlenecks</li> </ul>	<ul style="list-style-type: none"> <li>• High number of devices in the system suggests some reliability features may be needed to keep up long enough to sustain ingest</li> </ul>
Lenient	<ul style="list-style-type: none"> <li>• Can accept process failures, node outages, and thread failures as long as recovery brings data to a consistent state</li> <li>• Can reduce end-user query to support ingest processing</li> </ul>	<ul style="list-style-type: none"> <li>• Highly available, never-fail components are not needed</li> <li>• Neither are highly redundant components</li> </ul>	<ul style="list-style-type: none"> <li>• Highly redundant storage is not needed</li> <li>• Slower, less expensive devices may be appropriate for detection record storage</li> <li>• Interconnect failover through redundant paths not needed</li> </ul>

### 9.1.3 Storage Pooling Implications

The storage pool must grow over the life of the system. The vast majority of celestial object records will be created early during the operational period, and their numbers will remain fairly constant thereafter. However, detections arrive to the ODM in more-or-less a linear fashion over the life of the system. Rather than pre-configuring the ODM with the full quantity of storage that is likely to be filled only after two years of operation, storage should be added only as the volume of detection records increases. To minimize disruption to system operations, these storage increments must be added without reconfiguring processing or applications.

Detection records should be organized spatially, rather than by frame, to speed up object record maintenance. When updating an object record's aggregate properties, its time history of detections must be retrieved from the ODM. Fewer I/O operations are needed if those detections are co-located in storage. However, this need implies that the growth in storage, assuming the data are partitioned by regions on the sky, will not be evenly distributed over time. That ramification further implies a storage pooling approach that isolates storage from applications and processors.

The storage pool should be visible to multiple processors/applications for load balancing. At the same time, I/O activities to limit storage contention should be isolated. The storage pool must support storage partitioning to isolate the I/O activities of the other pools. It should also support striping across devices and controllers to improve throughput and reduce device contention.

The storage pool should be able to recover from errors or media failures with minimal disruption to operations. While the overall availability requirement on the ODM is relaxed for prototype deployment, frequent outages to replace and rebuild failed storage devices will tax the end-users' patience and introduce some risk of permanent data loss. Additionally, excessive outages may require reprocessing of published products, which could prevent the ODM from meeting the necessary ingest rates. Accordingly, some level of data protection is appropriate for the ODM.

Pooling of different types of storage is also desirable. Object records are accessed frequently, detection records less often, and 3 $\sigma$  difference sets rarely. Use of lower cost, higher density storage for less-accessed items will reduce the number of storage devices and the overall cost of the ODM.

A desirable storage pooling approach is to use external storage to allow incremental addition of capacity without disruption to applications or processors. This configuration would be able to aggregate and virtualize physical devices, further reducing disruption to other components of the ODM, while storage is reconfigured.

## 10 PS1 Prototype Design

Resource pooling provides design advantages since it is a strategy that relegates responsibilities to sets of resources capable of dividing and conquering a data management problem. Pools are not bound by the same scaling limitations as individual components. Application pooling, for example, can be extended by adding more processes to existing hosts, while widening the load balancing queue. Additionally, pools do not require homogeneous specification. Larger boxes might be added to a processing pool to concurrently service different sets of problems—such as the ODM's ingest correlation—while lesser hosts pool together to service query processing.

Resource pools can also be upgraded with technology refreshes without losing legacy investment. For example, storage pools can be upgraded by disk sizes, types, and speeds within the existing environment. The pooling strategy empowers design opportunities throughout the application architecture and data lifecycle.

The analysis of the Pan-STARRS PS1 ODM challenges clearly indicates an architectural strategy that strives to pool all three data management resources. Pooling of storage resources supports the large, fast, and dynamic nature of the system, allowing the storage to be provisioned incrementally. Pooling of processing resources can significantly reduce the resource requirements by allowing all processors to support any activity. Finally, pooling of application resources will greatly simplify query processing, can reduce storage requirements by reducing boundary data replication, and can significantly reduce I/O requirements.

Architectural pools provide design strategies that support a balanced architecture with a flexible, scalable, and optimized approach for Pan-STARRS’s PS1 Object Data Manager. The schematic design we have adopted for the PS1 ODM is shown in Figure 3. In this design, we have split the primary functions of the ODM—data ingest and query response—into autonomous cooperating subsystems. The data ingest for the ODM takes place in the “left brain” subsystem over the course of the monthly cycle during which the IPP publishes its products to the PSPS. While the ingest subsystem is performing those operations necessary to incorporate the new detections into the database and update celestial object records, all the data that has already been published and ingested into the ODM is available for astronomical queries via the “right brain” side of the system. When the ingest side has finished preparing the detections for input into the ODM and updating object records, the updated objects are copied to the right brain and the new detections are made accessible for queries. While the ingest processing is taking place on the left brain, we plan to run Published Data Clients (PDCs) on the right brain in addition to serving user queries. One use of a PDC will be to examine, and if necessary, refine detection-object associations made during the ingest processing. Any corrections the PDC might generate to these associations would be fed back to the left brain for updating in the next monthly update cycle. Other PDCs might perform astrometric analyses of the detections, compute photometric redshifts, etc. The PDC will also be a mechanism available to the database users; however only those PDCs that are part of the ODM will be able to update the database itself.

The ingest processing server will be handled by a single multi-processor Enterprise class server machine. For PS1 our baseline configuration for this subsystem is a server with 16 sockets populated with Itanium dual core processors. To allow for potential growth, the system will be expandable to a maximum of 32 sockets as a single server. The anticipated hardware will support 128 GB of RAM and 8 host bus adapter (HBA) ports connected to the data storage area network (SAN).

The initial deployment of query servers for PS1 will consist of two quad-socket servers using dual core Xeon processors. Each processor will have 16 GB of RAM and 2 HBAs.

### Preliminary Design Details: ODM Hardware Layout

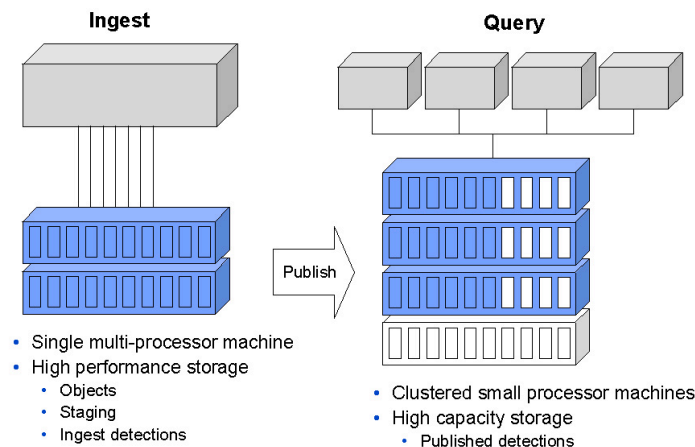


Fig 3. Preliminary Design Layout for the PS-1 ODM

The disk storage for the ODM will employ a pooled strategy, with both high performance storage (10-15K drives) for storing the celestial objects, and lower performance high-capacity disks for storing the published detections. At deployment, we plan to have 10 TB of storage allocated for the celestial objects and 100 TB of storage available to hold the detections. As indicated by the blank slots on the right side of Figure 3, the detection storage will not be fully populated when the system becomes operational, but can grow as the demand requires. Our current planning for the PS1 mission is to provide up to 420 TB of storage for the detections.

One final note about our design deserves comment with regard to the storage, namely that we will have a copy of the celestial objects available on both sides of the ODM “brain.” This is done so that the ingest preparation does not impact the query performance seen by the ODM users, and to provide a snapshot backup of the objects should one or the other side experience storage failures for the objects.

## 11 Scaling the PS1 Design to PS4

The PS1 system was originally proposed at a risk assessment and reduction activity. The ODM was definitely considered to be one of the major risks involved in Pan-STARRS by those recommending that PS1 be undertaken prior to building the full PS4 system. While PS1 is a quarter-scale system in comparison to PS4, its database will be a major leap over those currently representing the “state of the art” in astronomical databases, e.g., the 2MASS database or the Sloan Digital Sky Survey (SDSS). These systems have ~1-2 billion entries stored in roughly that same number of rows. We estimate PS1 will need to keep track of ~10 times more objects, and will ingest per month as many rows as currently contained in either the 2MASS or SDSS databases, resulting, at the end of the PS1 operations, in a database with over 40 times more rows than the current systems.

We are confident the design we are planning for PS1 will meet the requirements for the PS1 mission, both as a test bed for the database development effort and as a scientific tool for members of the PS1 science team as well as those of the PS1 Science Consortium being formed to support the PS1 science programs. Given that PS4 will need to load ~4 times more detections per second and maintain object information on perhaps ~2-4 times more celestial objects than PS1, if the PS1 design can’t scale up, it can certainly scale out by simply replicating the PS1 configuration 4 or more times. And, while it may seem straightforward, the primary challenge will be to do so in a way that constrains the total system costs as much as possible.

A major thrust of our design effort has been directed to the data ingest problem. Pan-STARRS will be the first major survey program to open up the study of the time-domain in astronomy on large scales. Other projects, such as the Large Synoptic Survey Telescope, will follow on this path. If we are unable to make these data available to users in a “reasonable” period of time without falling behind the data production at the telescope and the IPP, we have essentially failed to attain one of our primary goals. A fact of life, however, is that the *users* only see system performance from their perspective, i.e., how fast can it give me the information I asked for?

While the total user base of the PS1 Science Consortium is currently unknown, it should easily be sufficient to develop a profile of the usage patterns in the ODM. Our architectural design should allow for straightforward expansion on the query side of the ODM brain at very moderate cost should the initial hardware deployment be inadequate to meet the needs of the user community. Establishing how astronomers want to search the data is essential to taking the PS1 design to the next stage in that we can start to understand tradeoffs in use and design that drive the system cost.

## Acknowledgments

*The authors wish to thank contributors and reviewers of this paper, and our colleagues on the Pan-STARRS project. In particular, we appreciate the insights and suggestions offered Messrs. Christopher Argauer and Bernard C. Cotton.*