Glossary

Abstract factory A design pattern for instantiating an object whose class is not directly known to user of the abstract factory. *See* [GHJV95].

Access node A node that provides external interfaces to subscribers or other systems.

Ack Positive acknowledgment. In a request–response message pair, an ack indicates that the request succeeded.

Active processor The processor that is performing payload work in a HOT STANDBY or WARM STANDBY configuration.

Administration node A node that downloads data to other nodes and which provides craftspeople with a view of the system's behavior.

Alarm An indication given to craftspeople when a hardware or software failure or resource shortage has caused a service outage or degradation for a number of subscribers.

Application checkpointing Adding checkpointing software to applications in *ad hoc* manner. The software sends messages from an active processor to its standby to replicate objects on the standby.

Asynchronous messaging A form of messaging in which the sender's software continues to run rather than being blocked while waiting for a response. It requires the use of state machines. The opposite of synchronous messaging.

Robust Communications Software G. Utas
 ISBN: 0-470-85434-0 (HB)

Audit A background thread that looks for, and typically tries to correct, inconsistencies such as orphaned resources or corruptions in data structures.

Auxiliary data block A block of memory that is appended to an object to hold some data when not all of the object's data will fit into the data block provided by its OBJECT POOL.

Availability The proportion of time that a system is in service.

Babbling idiot An interface that is generating an excessive number of messages.

Blocked thread A thread that cannot run because it is waiting for a blocking operation to finish.

Blocking operation An operation that causes a thread to be scheduled out until the operation completes. Examples include sleeping, acquiring a semaphore, sending a synchronous message, reading a socket, or performing disk I/O.

Bug A software fault that leads to an error or failure.

Callback A function that a client provides to a service so that the service can later invoke the function to inform the client of an event that is of interest to the client. It is often implemented by having the client register an object in a REGISTRY owned by the service.

Capacity The amount of work that a single processor can perform.

Carrier A firm that operates a network of extreme systems.

Carrier-grade A system that provides the high levels of availability, reliability, capacity, and scalability required by a carrier.

Checkpointing Maintaining synchronization between two processors (an active and a standby) that run in HOT STANDBY or WARM STANDBY mode. It allows the standby processor to preserve work during a failover.

Cold standby A survivability strategy that sets aside one or more spare processors to take on the work of any in-service processor that fails.

Concrete factory An object that instantiates objects in a specific class that is not visible to users of the concrete factory. *See* [GHJV95].

Connectionless protocol A protocol in which each incoming message can be processed independently of previous messages. It consists of either datagrams (unacknowledged, unsolicited messages)

or request–response message pairs, and is therefore restricted to one or two messages per dialog.

Connection-oriented protocol A protocol that defines phases for connection setup, message exchange, and disconnection. A connection setup request creates a session that must maintain state information to process subsequent requests correctly. Unless the connection is rejected, each dialog consists of three messages (connect, connect ack, and disconnect) plus additional service requests between the connect ack and disconnect.

Context switch Switching execution from one task to another. A context switch can occur during a blocking operation or as the result of preemptive scheduling.

Control node A node that monitors the status of other nodes and initiates recovery actions when a node fails.

Cooperative scheduling A scheduling policy in which threads, rather than the scheduler, determine when context switches occur. The threads generally run to completion. The opposite of preemptive scheduling. *See* [PONT01].

Craftspeople The people who administer and operate a system or network.

Critical region A section of software which accesses data that is also accessed by another thread. A semaphore must protect a critical region because a context switch could cause one thread to access the data while another thread is in the process of modifying it.

Cyclic scheduling A scheduling policy in which each thread runs at a regular interval. Often used in hard real-time systems.

Daemon A task which is created during system initialization and that lives forever, or at least until a restart occurs.

Deadlock A situation in which a group of threads remains permanently blocked because of a circular dependency among blocking operations. A deadlock occurs if, for example, two threads send each other synchronous messages at about the same time.

Dialog A sequence of interdependent messages exchanged in a protocol.

Distribution Dividing work among multiple processors in order to increase a system's overall capacity or survivability.

Dynamic data Data that changes frequently enough to disqualify it from residing in WRITE-PROTECTED MEMORY.

Embedded object An object that resides within another object and whose lifetime is therefore bounded by that of the outer object.

Embedded system A system that is dedicated to a specific purpose.

Error A deviation from specifications (incorrect operation) caused by a fault.

Escalating restarts A strategy for returning a node to service as quickly as possible by only partially reinitializing its software. If the reinitialization fails to keep the node in service, the scope of the next reinitialization increases.

Event An input to a state machine, causing the state machine to perform work.

Event handler Software that performs a state machine's work in reaction to an event. It is usually selected based on the state machine's current state.

Executable *See* Process.

Extreme system A system whose design is constrained by the carrier-grade forces of availability, reliability, capacity, and scalability.

Faction A set of threads that share a minimum percentage of CPU time. Under PROPORTIONAL SCHEDULING, factions replace thread priorities.

Failover Switching work from one processor to another under HOT STANDBY, WARM STANDBY, or COLD STANDBY.

Failure An error that causes the loss of work or renders a component incapable of performing work.

Fault The root cause of an error or failure.

Five-nines A measure of availability in which a system is out of service for no more than roughly 5 minutes per year.

Function tracer A trace tool that captures the sequence of function calls executed by software and displays them in an indented format to show nesting.

Gobbler Software that uses an excessive amount of some system resource.

Half-object plus protocol (HOPP) A design pattern that fully or partially replicates an object in other processors to improve their capacity. *See* [MESZ95].

Half-sync/half-async A design pattern that separates I/O (receiving messages) from the work performed in reaction to those messages. One or more I/O threads queue incoming messages on work queues that are serviced by one or more invoker threads. *See* [SCH96a].

Hard-deadline scheduling The need to run a thread close to, but no later than, a specific time. Often a requirement in hard real-time systems.

Hard real-time system A system that *must* perform work within a specified time to perform correctly.

Heartbeating Sending an 'I'm alive' message to the entity that manages a component to prevent it from declaring the component to be out of service.

Heterogeneous distribution Distributing work by assigning different functions to different processors.

Hierarchical distribution Distributing work by assigning lower level, CPU-intensive functions to some processors (such as access nodes), and higher-level, more complex work to other processors (such as service nodes).

Hitless failover A failover under WARM STANDBY or HOT STANDBY. It preserves work in the standby processor to avoid disrupting service.

Hitless patching Installing a software patch without disrupting service.

Hitless upgrade Installing a new software release without disrupting service.

Homogeneous distribution Distributing work by assigning different subscribers to different processors so that all processors perform the same functions.

HOPP *See* HALF-OBJECT PLUS PROTOCOL.

Hot standby A survivability strategy that provides hardware checkpointing between two processors by verifying synchronization after each CPU instruction.

Input handler Application-specific software that an I/O thread invokes to identify the work queue in which a message should be placed. The appropriate input handler is selected based on the protocol of the incoming message.

Invoker thread A thread that takes incoming messages from a queue filled by an I/O thread and invokes application objects to process these messages.

I/O level Software that is responsible for performing I/O, such as `tnettask` and I/O threads.

I/O thread A thread that receives messages and places them on a work queue for later processing by an invoker thread.

IP port registry A REGISTRY that selects an input handler based on the IP port on which a message arrived.

ISR (interrupt service routine) Software that handles a processor interrupt, such as an I/O interrupt or clock interrupt.

Kernel mode A memory protection strategy in which operating system data structures are made inaccessible to threads except during system calls.

Latency The delay between receiving a request and sending a response, which is primarily determined by how long a request waits in a queue before it is processed.

Leaky bucket counter A design pattern that detects when a certain number of events occur within a certain timeframe. It is often used to track faults so that a system will neither overreact to intermittent, expected faults nor spend too much time handling a flurry of faults. *See* [GAM96].

Load sharing A survivability strategy in which two or more processors share work, such that the remaining processors continue in service if any of them fail.

Locked thread A thread that has acquired the run-to-completion lock to run unpreemptably with respect to other threads that also contend for this lock.

Log An event that is brought to the attention of craftspeople. If the event has caused a service degradation or outage, it also raises an alarm.

Maintenance Autonomous or manual procedures that implement fault detection, isolation, notification, correction, and recovery.

Memory checkpointing Sending modified pages from an active processor to its standby to maintain synchronization.

Message An input that belongs to a specific protocol. Its contents consist of a signal and zero or more parameters.

Message tracer A trace tool that captures entire messages.

Morphing *See* OBJECT MORPHING.

n + *m* **sparing** *See* COLD STANDBY.

Nack Negative acknowledgment. In a request–response message pair, a nack indicates that the request failed.

Network thread A thread that implements a messaging protocol, such as an IP stack.

Node A processor in a distributed system. In HOT STANDBY or WARM STANDBY, it often refers to a pair of processors in an active–standby configuration.

NUC (non-upward compatible) Changing an interface in a way that is not transparent to users of the interface. A NUC change to a class forces its users, at a minimum, to be recompiled. A NUC change to a protocol forces all users of the protocol to upgrade their software.

OAM&P Operations, administration, maintenance, and provisioning.

`Object` class The class from which all nontrivial objects ultimately derive. It defines functions that all objects should support.

Object checkpointing Providing a framework for APPLICATION CHECKPOINTING.

Object data block A block of memory allocated by an OBJECT POOL for housing an object created at run time.

Object morphing Changing an object's class at run time.

Object nullification Resetting some or all of an object's data (including its `vptr`) to improve the likelihood of detecting stale accesses to the object.

Object pool An object management strategy that pre-allocates object data blocks in a pool to avoid use of the heap at run time. A class that uses an object pool overrides operators `new` and `delete` to obtain blocks from, and return them to, the pool.

Object reformatting Reformatting data to the schema of a new release to prepare for a software upgrade, or converting checkpointed objects serialized in one release to their format in a new release.

Object template Using a block-copy operation to initialize an object quickly from a pre-allocated instance of that object.

Operational measurements Usage statistics presented to craftspeople to allow them to engineer a system and evaluate its throughput.

Operations Generating logs, alarms, and operational measurements for craftspeople to inform them of a system's behavior.

Operator *See* Carrier.

Outage A failure that causes a loss of service for some or all subscribers.

Overload control A strategy that prevents thrashing by prioritizing, discarding, or throttling work when a processor receives more work than it can handle.

Parameter An argument passed in a message.

Patch Software that is installed to fix a bug.

Payload work Work performed by services.

Placement `new` A version of operator `new` that constructs an object at a specific memory address.

`PooledObject` A subclass of `Object` that uses an OBJECT POOL to manage its objects.

Polymorphic factory An ABSTRACT FACTORY that delegates object instantiation to a CONCRETE FACTORY that resides in a REGISTRY.

POSIX (portable operating system interface) A standard that promotes software portability by defining a uniform set of operating system capabilities.

Preemptive scheduling A scheduling policy in which a context switch occurs as soon as a higher priority thread is ready to run or, under round-robin scheduling, when a thread's timeslice expires and another thread of the same priority is ready to run.

Primordial thread The first thread, the one that executes `main`.

Priority inheritance Resolving priority inversion by temporarily raising a thread's priority when it owns a resource that is blocking a higher priority thread.

Priority inversion A situation in which a higher priority thread cannot run because it is blocked by a lower priority thread. For example, the higher priority thread might be waiting for a semaphore currently owned by the lower priority thread, or it might be waiting for the lower priority thread to respond to a synchronous message.

Priority scheduling A scheduling policy in which a context switch selects the highest priority thread as the next one to run.

Process A thread or group of threads that run in a user space.

Programming model A set of techniques followed by all of the software in a system. Whereas a system's architecture primarily addresses its interfaces, its programming model primarily addresses its implementation.

`ProtectedObject` A subclass of `Object` that creates objects in WRITE-PROTECTED MEMORY.

Proportional scheduling A scheduling policy that assigns threads to factions and guarantees each faction a minimum percentage of CPU time. The opposite of priority scheduling.

Protocol A set of signals and parameters that are used to construct messages, along with rules that define the order for sending and receiving signals and the parameters that are mandatory or optional for each signal.

Protocol backward compatibility Evolving a protocol so that a processor running a new software release can communicate with a processor running a previous software release.

Provisioning Configuring a system by populating it with data.

Quasi-singleton A class whose objects usually behave as SINGLETONS but which makes provision for allocating another instance of an object when the current object is still in use.

Race condition A situation where software must handle the near-simultaneous occurrence of two events to avoid a bug. Examples include critical regions, deadlocks, and two processing contexts sending each other requests at about the same time.

Reactive system A system that responds to messages arriving from external sources.

Real-time system *See* Hard real-time system *and* Soft real-time system.

Registry An object that houses a set of polymorphic objects and which uses an identifier to select the appropriate object when delegating work polymorphically.

Reliability The percentage of time that a system operates free of errors.

Remote procedure call (RPC) A function call that sends a message, usually synchronously, in which case its thread blocks until it receives a response.

Restart Reinitializing a processor to recover from a software error.

Robustness The ability to maintain availability and reliability despite the occurrence of hardware and software faults.

Rolling upgrade Installing a new software release in a distributed system one processor at a time.

Round-robin scheduling A scheduling policy in which threads of the same priority receive a timeslice to perform their work, with a context switch occurring at the end of the timeslice if another thread of equal priority is ready to run.

RPC *See* Remote procedure call.

Run to completion A strategy that reduces critical region bugs by running each thread locked to suppress preemptive scheduling. *See* [DEBR95].

Run-to-completion cluster A set of state machines that collaborate by exchanging high-priority messages. This prevents members of the cluster from having to deal with messages that could arrive from outside the cluster during transient states which arise from their collaboration. *See* [UTAS01].

Run-to-completion lock A global semaphore which must be acquired by a thread that wishes to run to completion. It prevents preemptive scheduling among the threads that acquire it.

Run-to-completion timeout A sanity timeout that occurs when a thread runs locked too long.

Safety net Catching all exceptions and signals in a thread's entry function in order to clean up the thread's current work and reenter or recreate the thread.

Scalability The ability to increase a system's overall capacity by adding more processors.

Service Software that implements a capability used by subscribers.

Service node A node that primarily runs payload work (services).

Session A group of objects that implements an individual dialog in a connection-oriented protocol.

Session processing The processing of connection-oriented protocols.

`SharedObject` A subclass of `Object` that creates objects in a global shared memory segment.

Signal 1. A message type in a protocol. 2. An event that is received by a thread, usually to flag a software error that is fatal unless caught by a signal handler.

Signal handler A callback that is invoked when a thread receives a signal.

Singleton An object that is the only instance of its class. *See* [GHJV95].

SMP *See* SYMMETRIC MULTI-PROCESSING.

Soft real-time system A system that must perform work within a specified time to provide an acceptable response time.

Spinlock A loop that obtains a lock by polling. Only useful in SMP systems.

Stack overflow A fatal error that occurs when a thread overruns its stack, usually by nesting function calls too deeply or by declaring too many large local variables.

Stack short-circuiting A technique which improves capacity by bypassing the lower layers of a protocol stack when sending an intraprocessor message.

Stack trace The chain of function calls that appears in a thread's stack, starting with the thread's entry function and ending at the function that is currently running. It may also include each function's arguments and local variables.

Standby processor The processor that can immediately take over if the active processor fails in a HOT STANDBY or WARM STANDBY configuration.

State An object that defines how far a state machine has progressed in handling a transaction or session. It selects the event handler to invoke when the state machine receives an event.

Stateful Software that must use state machines to handle a sequence of messages in a connection-oriented protocol.

Stateless Software which need not use state machines because it can process each incoming message independently of previous messages.

State machine A set of states, events, and event handlers which implement some capability. A state machine is required to support a connection-oriented protocol.

Static data Data which changes sufficiently infrequently for it to reside in WRITE-PROTECTED MEMORY.

Subscriber An end user of a system, often a customer of a carrier.

Subscriber profile The data required to support a subscriber, such a list of the services that the subscriber may use.

Survivability The ability of a system to remain in service when hardware or software failures occur.

Synchronous messaging A form of messaging in which the sender's thread is blocked while waiting for a response. The opposite of asynchronous messaging.

Symmetric multi-processing (SMP) A hardware configuration in which multiple processors share a common memory.

System initialization Running the software that brings a processor to the point where it is ready to handle payload work.

Task A term that refers to a process or a thread when a concept applies to both of them.

Thread A sequence of instructions that executes on its own stack. A thread shares its address space with other threads that run under the same process.

`Thread Class` A WRAPPER FACADE [POSA00] for native threads which provides THREAD-SPECIFIC STORAGE [POSA00] and supports various extreme software techniques.

Thread pool A group of threads which perform the same type of work when it requires the use of blocking operations. Using multiple threads reduces latency because some (or all) of the work is performed concurrently.

Thread starvation Under priority scheduling, a situation in which lower priority threads are prevented from running because higher priority threads are using all the CPU time.

Tick The time between two clock interrupts that cause the scheduler to run. The length of a tick is system specific. A timeslice in a round-robin scheduler consists of a small number of ticks.

Timeslice Under round-robin scheduling, the number of ticks that the scheduler allows a thread to run before preempting it to run another thread of equal priority.

Timewheel A circular buffer whose entries are visited at regular time intervals to perform some function, such as scheduling threads or servicing timer requests.

TLV (type–length–value) **message** A message that encodes each parameter using a parameter identifier (type), length, and contents (value).

Tracepoint debugger A debugger that collects data at specified instructions and then allows the system to resume execution instead of halting.

Trampler Software that corrupts memory by writing through an invalid or stale pointer or an out-of-bounds array index.

Transaction The work performed in response to an incoming message.

Transaction processing The processing of connectionless protocols.

Trap A signal or C++ exception.

Upgrade Installing a new software release as opposed to a patch.

User spaces A memory protection strategy in which each process runs in a memory segment that is inaccessible to other processes.

Virtual synchrony A strategy that implements WARM STANDBY by passing each input to both the active and standby processors and verifying synchronization by comparing their outputs.

`vptr` (virtual function pointer) A hidden data member of each object that defines or inherits virtual functions. It references an array which contains a pointer to the correct instance of each virtual function that is associated with the object's class.

Warm standby A survivability strategy in which software provides checkpointing between an active and a standby processor.

Watchdog A timer whose expiry is treated as a fatal error. *See* [PONT01].

Worker thread A member of a THREAD POOL.

Write-protected memory A memory protection strategy that places critical data in a memory segment that can be read by all threads but that can only be modified after performing an unprotection operation.