Digital Game Systems Corp.

slotengine®

features overview

slotengine®

- Today's technology, designed with modern computing principles; no restrictions due to legacy technology debt; designed with open-source solutions
- Unlimited flexibility: Game studios independent of platform team: 100% control over games. Platform team can grow core feature set, and maintain product without impact to studios
- Unlimited scalability: No monolithic structures dictating deployment and performance — install on right-sized equipment and scale infinitely as needed
- Superior quality and reliability: With micro-services web architecture there are no processes that crash, no memory-leaks, no conflicting threads, no safety-reboots, no service windows. 24/7/365 operations.
- Small platform footprint and small maintenance: control costs in D&D and operations, maintaining and operating the platform
- Easy to learn, easy to work with: small learning curve, line by line visual debugging even at the framework level, game developers have full understanding of source — never black-boxed.
- Cost efficiency and future-proofing: open-source technologies have zero cost supporting software (OS, database, tools, utilities) and ever-evolving solutions from worldwide resources
- www.slotengine.com

vs. Legacy Platforms

- Legacy solutions are often decades old, proprietary code patched over time; a mix of many technologies and visions of many architects. The solution is no longer clearly understood across the organization, the solution architects are not available. Often, the technology debt makes platform evolution slow, costly and of deteriorating quality over time.
- Limited flexibility: Game studios are dependent on features platform currently supports. Game studios are unable to design a game not directly supported by the platform. Platform team are often involved in helping studios author a game.
- Limited scalability: Monolithic structures prescribe a specific deployment installation — and present a bottleneck, with limited scalability: typically, limited to running on a larger (more RAM, faster clock) server
- Limited quality and reliability: With legacy monolithic architecture key processes are designed to run in server memory, and over time experience memory-leaks and crashes, require occasional reboot and require specific down-time for service-window. Their black-boxing limits developers' understanding of the whole architecture, and the quality of overall code suffers.
- Large platform footprint and sizeable maintenance: visible costs in D&D and operations, for maintaining and operating platform
- Complex technology with a learning curve; visual debugging impossible since key "engine" components are black-boxed. In many installations studio developers do not have full understanding of the code running their game.
- Cost inefficiency and solution locked-in: proprietary technologies demand specific talent; vendor solutions such as Windows and MSSQL have costs supporting software (OS, database, tools, utilities) and forever lock studio to a single vendor solution and platform/OS "religion"

slotengine®

- SlotEngine Remote Game Server is a purpose-built games platform that encapsulates and runs a portfolio of server-side casino games
- SlotEngine RGS must be integrated to a host casino platform
- Server-side games are authored to specific SlotEngine API and run under the SlotEngine open framework and other open-source OS components
- Client-Side games can be implemented using any technology, as required by the authoring Game Studio. They may be mobile device Apps (iOS or Android), or web apps written in Javascript under HTML5, or native/downloadable C++ modules. They simply need to use the SlotEngine API.
- SlotEngine and the server-side games run under a *Linux environment deployed in a private or public cloud eco-system
  - (**while the solution has been deployed and certified under CentOS Linux, and AWS Linux 2, there is no code dependency on Linux. The SlotEngine framework may be deployed anywhere PHP-FPM is available, including Windows)*

the solution stack

- HAProxy, high-availability open-source load balancer software
- NGINX, high-performance open-source web server software
- PHP-FPM, high-performance open-source PHP processor
- Redis, high-performance open-source in-memory database/cache
- MySQL, or Percona (open-source), high performance community release database
- SlotEngine API & Framework, licensed object-oriented class library of code (PHP7), implementing the RGS functionality
- Game library (studio's own) of Game-server software, representing your games
- Linux CentOS (open-source), or macOS, or RedHat or AWS Linux 2

Deployment

- Micro-service architecture and multi-tier server-side decoupling allow for infinite scaling, fault tolerance and high availability.
- deployment can be scaled horizontally across any number of processing servers, as operational throughput grows and demands
- deployment can be scaled vertically across multiple tiers, separating processing, cache and database tiers into separate servers, as operational load demands. Moreover, the processing tier can be configured to handle multiple parallel sessions, only limited by RAM and number of CPU or CPU cores
- The entire solution can fit and operate on a laptop computer, ideal for game development, testing, and zero-fail demos (most trade shows offer horrific networking conditions)
- For massive-scale computing, the solution may also be deployed across any popular cloud solution (AWS, Azure, Google, IBM for example) for computing, static/archiving/backup storage, database services, caching, load-balancing and network distribution services
- The solution may also be deployed in a hybrid scenario, where content distribution is handled by a cloud provider, software installation/computing is housed in a private server facility, and the database services are housed in a multi-tenanted special purpose server facility — as approved by relevant gaming jurisdiction.

Performance

- Performance for SlotEngine is only limited by the deployment (RAM, CPU, Disk space). In other words, it is theoretically unlimited.
- Due to its micro-services architecture, and stateless transactional design, processes require minimal CPU time to run
- When no one is playing, *nothing is running. When a burst of transaction requests occurs, they are distributed across all available servers for completion. When they complete, all processing on those servers terminates.
- Since SlotEngine does not have a monolithic module to process requests in an installation, there is no processing bottleneck, no memory-leaks, no multiple thread complexities and problems.
- SlotEngine relies on NGINX™ webserver/proxy and PHP-FPM open-source technology to process transaction requests. This technology can handle massive computing scale needs, used by companies such as Snapchat, Slack, Facebook, Twitter, among many others…
- Typical installation for a casino client may include HA (high availability) configuration with a cloud-based DNS configuration, and a load balancer, and private dual processing servers, dual cache servers and dual database servers, which will have no issues in handling 500 transactions per second, or about 30,000 games per minute.
- Massive computing platform can handle massive performance demands: with proper resources configuration SlotEngine (NGINX, HAPROXY, PHP-FPM, REDIS and MYSQL) **can handle millions of gaming transactions:**
  - www.scalingphpbook.com *"Hey! I'm Steve Corona. I spent 5 years scaling Twitpic to handle 60 million users and 20 billion HTTP requests per month"*

API

- Open-platform, web micro-services, 100% extensible
- Game design is entirely in the Studio's hands, without reliance on changes required by from a platform team
- Multiple SlotEngine API families:
  - API for connecting Game Clients to Game Servers
  - Framework and Templates for writing Game Server Software
  - API for SlotEngine Administration
  - API for legacy Transfer Wallet Casino Platform Integration
  - Seamless integration with Casino Platforms using Player And Wallet

Casino Platform

- Operators run own (procured) casino platforms (NYX, GAN, Playtech, etc.)
- SlotEngine is a game publisher's game platform that integrates with Casino Platforms, and delivers its game portfolios to the Operator.
- SlotEngine and its games rely on Casino platform for player account management, loyalty, restrictions, player limits, account funding, and similar functions
- Casino relies on SlotEngine to deliver and play a game publisher's portfolios of games
- SlotEngine integrates to casino platform using the Player Account And Player Wallet API and concepts, for seamless (transactional, bet by bet) integration
- To support legacy casino platforms, SlotEngine can also integrate using its Transfer Wallet API and concept.

Real Money Compliance

- Designed for casino games accounting, reporting and auditing
- Full transaction history and “true” game replay— administrators can watch the game reload and replay exactly as the player experienced it
- SlotEngine software has been GLI-19 certified
- Responsible-gaming aware
- Geo-fencing aware
- SlotEngine’s Industry standard software RNG has been certified for UK and European gaming jurisdictions; RNG standard recognized worldwide

Game Client Software

- Modern Game clients are authored in Javascript under HTML5 or as native mobile Apps.
- Studios use most appropriate technology for widest possible channel coverage, and efficiency in transaction costs:
  - Mobile Web has no costs but requires own distribution points
  - Apple Store or Google Play provides well-known distribution point, but costs ~30% of gross sales
- Regardless of Studio's technology choice, clients use SlotEngine API, a set of web-services, to conduct game play.
- Communication from Client to Server occurs over HTTPS using industry-standard strong encryption
- Data exchanged between game clients and the RGS game servers in delivered in JSON-formatted messages

Game Server Software

- Under the SlotEngine Architecture, each Game Client software has corresponding secure Game Server software
- It is a standard set of web micro-services that use the SlotEngine framework for generic gaming processing, and
- use private, game-specific server-side logic to perform game play unique to this game design
- Web Micro-services (API) includes: Bet change, Config get, Denom change, Play, and Session Create

Hybrid multi-tenancy

- SlotEngine RGS is designed for multi-tenancy
- Same software installation servers can support multiple casinos
- Same database servers can support multiple casinos
- Hybrid models are supported, where different installations (due to jurisdictional restrictions) under separate certifications are connected to the same/shared database servers

Multi-currency

- Designed for Multi-currency operations
- Social “CREDITS” treated as another type of currency, if running in Social operations (not RMG)
- All accounting and reports are organized by currency
- Currency definition and use is entirely data-driven (a configuration update)

Comp handling

- SlotEngine recognizes comps as a key tool for implementing a player loyalty scheme
- Comps may be granted to specific individuals based on specific game they are playing
- When comp balance is detected, comps are bet first
- When comp balance is too small to cover a bet, bet is made from available comps, plus real money
- Comps are granted by secure API, triggered by casino platform's marketing scheme and relevant tools
- All accounting and game reports account comps separately for proper gaming reconciliation

SlotEngine Administration

- Operator's administration portal handles
  - Operations Status and Installation Validation
  - User management
  - Game Management
  - Comp Management
  - Session Management (reviews, replays)
  - Reporting (games, players, performance, casino, accounting, logs)
  - Messaging (broadcasting to player, game or casino)

Roadmap

- Multi-player Games
- Global Linked Progressives
- Integration to 3rd party platforms
- Adoption by other studios

In summary

- SlotEngine is today's technology, without technology debt
- It is highly flexible, not restricting game designers to black-box design
- It can scale without limitations, supporting needs for massive computing
- It offers superior quality and reliability: 24/7/365 operations
- It has a small footprint requiring minimal support
- It is easy to learn and adopt: game developers have a full understanding of the complete solution
- It is highly cost efficient and future-proofed: open-source technologies have zero cost, are ever-evolving solutions from worldwide resources, and present a choice of multiple vendor solutions
- Thank you!