DISTRIBUTED COMPUTING is a DECENTRALIZED LOAD BALANCING

S.MOHAN

ASSISTANT PROFESSOR OF COMPUTER SCIENCE AND ENGINEERING,

V.S.B. COLLEGE OF ENGINEERING TECHNICAL CAMPUS,

KINATHUKADAVU, COIMBATORE, TAMIL NADU ,INDIA-642 109

ABSTRACT:

Distributed computing is a decentralized computing model where multiple independent nodes coordinate to the process task .while load balancing distributed traffic , distributed computing act as underlying infrastructure that natively distributed computational local without a central point of control prioritized scalability , fault tolerance and resource aggregation . it has no single point failure . instead of relying single central local balance . which can become a bottleneck or fail completely . distributed system empower each individual nodes to makes work load division . so it is peer-to-peer(p2p) cooperate in distributed architecture (e.g. block chain, edge network or grid computing nodes commit with neighbor to share resources , migrate task and balance network loads consensus protocol decentralized system use algorithm the RAFT, PAXOS or GOSSIP protocol) to main the synchronous state access the network , inherently distributing traffic and data consistency . because it has randomized polling nodes recently querying other nodes in the network to lightly load sever for task delegation we will use GOSSIP portal for nodes periodically and recently exchange load statistics with their peer . creating the grateful balanced global state over under loaded nodes pro actively search for over loaded nodes (receiver-waited) or over loaded nodes search for available capacity (sender- immediate) to migrate task without centralized direction if a nodes goes off-line, other peer can takes over the work load . this process close to the edge or near specific node reducing network delay , nodes globally synchronized state and share the process work load the decentralized SWAM balance the upload and download burden among peers leverage processing task are systemized and handle concurrently across computing clusters.

KEYWORDS : API, GOSSIP,RAFT,host.

INTRODUTION:

Distributed computing connect multiple networked computer to work collaborate to keep the system rely efficiently , work load are dynamically spread across all active nodes a process known as load balancing . which traditional system are a central server to route traffic a decentralized load balancing approach enable individual nodes to communicate and menage work load automatically . instead of relying a single “MASTER” balancer that could between a bottleneck or point of failure decentralized system use peer-to-peer protocol node connect directly to share processing power and balancing traffic. Decentralized load balancing in distributed system is a technique that distributed load among the nodes by considering the processing capacity of node this approach is particularly useful in heterogeneous system when nodes how different process capacity . the load balancing algorithm uses diffusion technique to distributed the load from heavily located node to lightly local node ensuring that the distributed of the legal is uniform after Appling the load balancing algorithms .this method is connected mashing the performance and reliability of the system especially in scenario when the load among the nodes is not uniforms. In a distributed system multiple server , nodes or computing resources work together to havable request and performance task load balancing ensure that their these work load are evenly distributed presenting the bottlenecks and improving are all system efficiently . it acts as a “traffic cop” directing request to the most appropriate server based an connect load availability and predefine rules.

LOAD BALANCING MEANS:

Load balancing is the critical technique employed across various domain to ensure optional performance reliability and scalability system . it is widely used in the following areas.

WEB APPLICATION:

load balancing is essential for distributed managing traffic across multiple server HOSTING web application . this ensure that no single server because overwhelmed improving the response.

CLOUD COMPUTING :

In cloud environment load balance distributed work load across virtual machine or continues . those enable historical scaling , where resuming can be added or removed dynamically based on traffic demands.

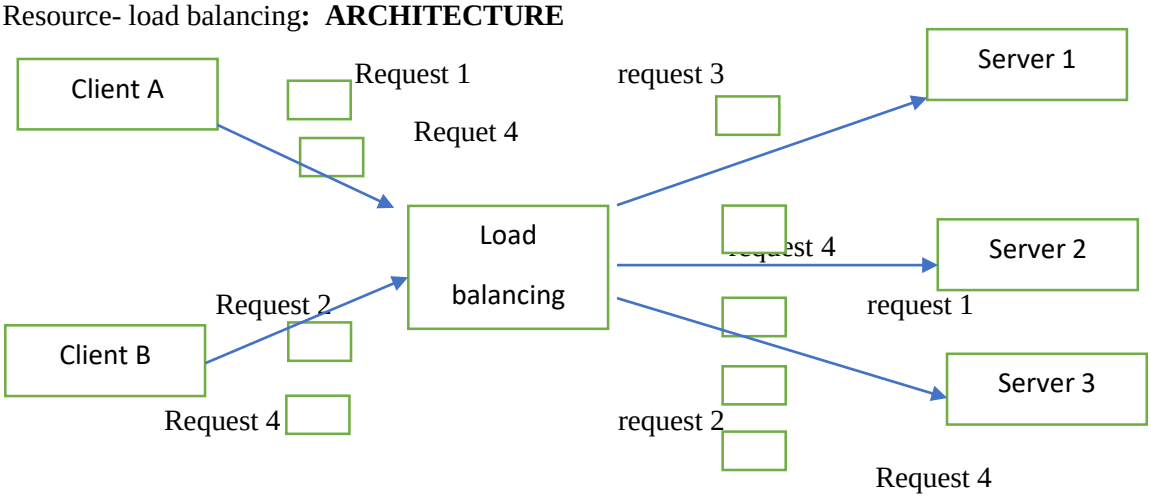
DATA CENTER:

load balancing is heavily utilized in data center to manage traffic within large networks . it uses efficient utilization if server resources and present bottleneck.

GLOBAL SERVER LOAD BALANCING:(GSLB)

For application with global user bases ,load balancing distributed traffic across server located n a different geographical region .this minimize latency and optimization performance for user world wide.

Resource- load balancing**: ARCHITECTURE**



CENTRALIZED LOAD BALANCING:

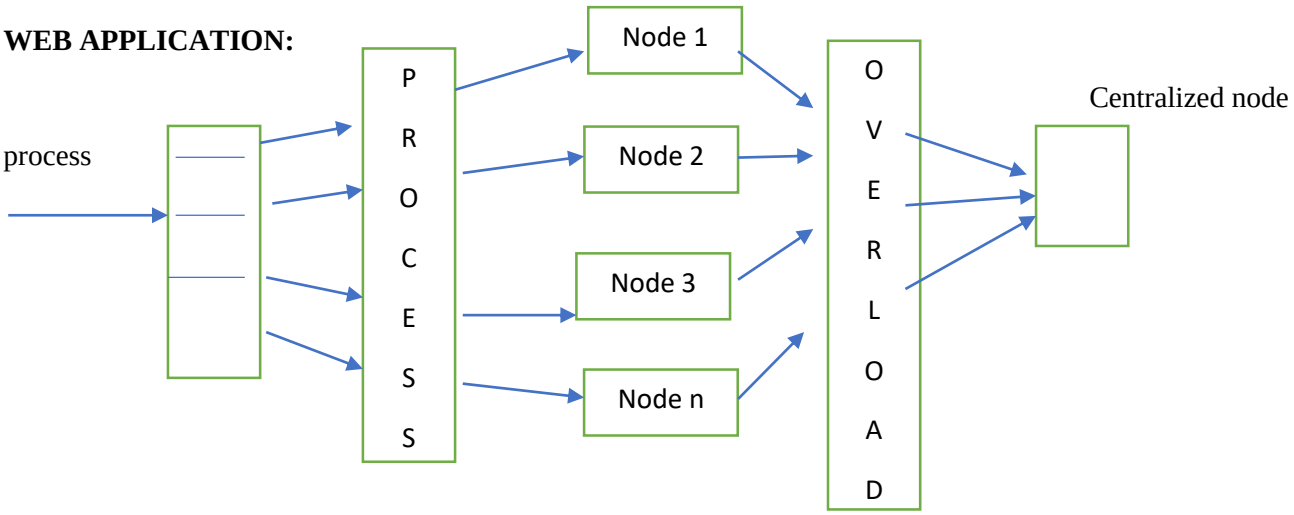
it refer to the management of the load distribution across the multiple server in a centralized name. this approach is critical for maintaining system reloadability and performance . it introduce load balance to distribute in coming network traffic evenly among server .precenting any single server from becoming a overwhelmed .this distribution helps to ensure that the system can handle in crashed request without perform degradation . this maintaining is consistence is and responsive source . centralized local balancing is essential for cloud based architecture it is allow for seamless traffic management at high availability across the multiple region.

Centralized load balancing is a method where a single load balance manage and distributed increasing traffic across the multiple server from a central point centralized load balance prove a single architecture point for distributing networks offering simplified management , optimized resource use and integrated security. However it required careful design to avoid bottleneck and ensure high availability often high reducing or cloud based distributed implementation.

CLOUD ENVIRONMENT:

Centralized local balance are commonly used in clou platforms like google cloud , where a single load balance can manage traffic across multiple region and project , processing high accessibility and low latency

WEB APPLICATION:



E-commerce site or high -sale used server often used centralized load balancing to ensure the available performs for the during peak for traffic periodic.

DISTRIBUTED COMPUTING:

This is over achieve the system architecture . it in values multiple independents compute node connected over the networks that works together as a single system to solve a problem or HOST as a applications.

LOAD BALANCING:

this is a specific function applied to distributed system has many server a load balance act as a "TRAFFIC COP" sitting in front of the server to intercept use request and route then to the specific machine that is best equipped to help them.

CENTRALIZE vs DECENTRALIZED load balancing:

CENTRALIZED a single distributed device (or CLUSTER) devices when all traffic goes it act as the single entry points.

DECENTRALIZED: in derived node in the system commits with each others to independently figure out how to share task and route traffic without a central activity.

DECENTRALIZED LOAD BALANCING:

Decentralized load balancing eliminate the central departures in distributed computing instead of individual node is load state observation and peer-to-peer in variation to the autonomously distributed computability or communication task . this self-organization approach provide high scalability , resilience to node failure and eliminate performance bottlenecks.

WORKS:

Unlike decentralized architecture that rely on a single load balancer decentralized modern engine ever node to make selected device based on it own view of the network . this process release on these key mechanism.

STATE INFRASTRUCTURE EXCHANGE:

node periodically exchange information equally connect load (eg CPU utilize, MEMORY,ACTIVE QUEUE,)with neighboring nodes

Initiation policy :

Load distributed is triggered based on specific condition

Sender- initiative:

Heavily load nodes actively search for lightly loaded node to off-loaded task.

Receiver- initiative:

Under-utilize node actively poll neighboring nodes request pending work load.

COMMON DECENTRALIZED ALGORITHMS:

Nodes use self-organization strategies to converge towards global located uniformly without central over rights.

GOSSIP protocol:

Node reducing exchange load start with a few peer , over time, this epitomic style communication spread state infrastructure access the entire networks allowing all nodes to route jobs to the least load areas.

DIFFUSION ALGORITHMS:

heavily loaded nodes celebrates discrepancy between this load and that of the neighbor . thus offered proportion amount of excess work to neighbor nodes with lower work loads.

GAME - THEORATICAL APPRPACH:

Nodes act as a individual agents aim to manage then own task execute time or cost which materially results in a Nash equilibrium that balance the global load .

MULTI-AGENT REINFORCEMENT LEARNING:

nides learn from past running device to optimize future task distribution based on return latency and changing the traffic [pattern.

ADVANTAGEIOS AND TRADE OFF:

Pros:

No single point of failure if one node goes off-line the rest of the decentralized system container to functioned re decentralized work loads.

High scalability scale efficient as new node are added without requiring update to a central hardware and software balanced.

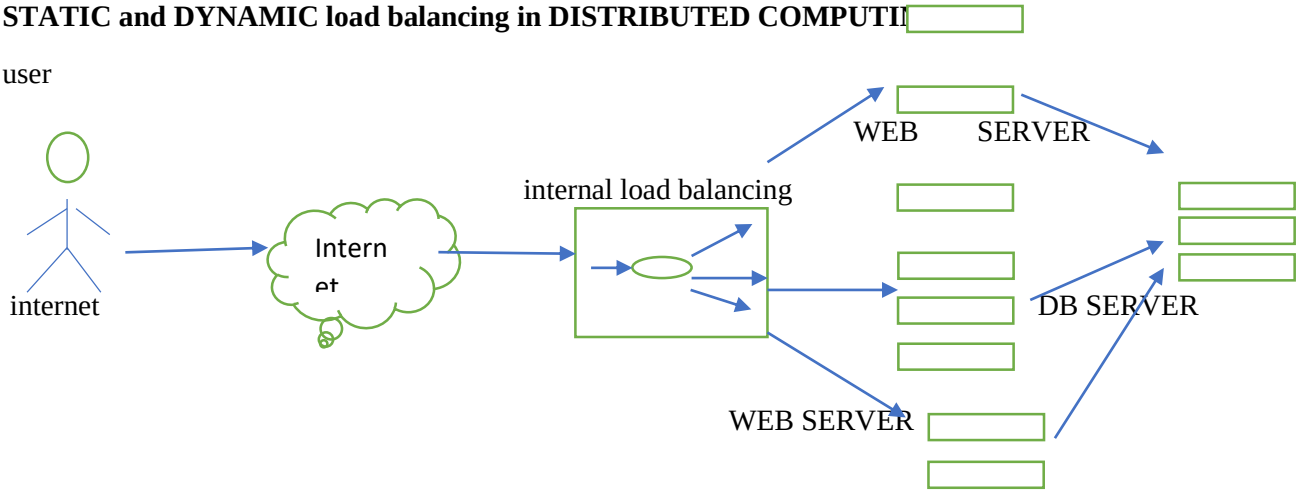
Cons:

Delayed state visibility because node rely on local or exchanged information , they might maker decision using slightly outdated state data.

Over head and migration cost:

The contrast communication required to update node state (often called “GOSSIP over head”) can connect network bandwidth and import over all performance if not turned correctly.

STATIC and DYNAMIC load balancing in DISTRIBUTED COMPUTI



load balancing is a distributed computing eventually distributed network traffic or computational task across multiple

sever . static load balancing uses fixed ,pre- determine rules without corresponding server condition . dynamic load balancing adapt in real-time continuously monitoring server least and work load to optimize performance.

STATIC LOAD BALANCING:

Static algorithm make roughly division based solely on prior knowledge of the system (eg sever capacity) regardless of connect traffic or server load .

How it works: task are assigned according to pre-scheduled.

Advantageous: it simples to import low computation overhead and work effectively well with periodically work load

Disadvantageous:

It lacks of adaptability if a server goes off-line or expenses a sudden spite the algorithms continues sending traffic to it.

Common algorithms:

Round- robin :distributed request sequentially in a fixed , respective order

Weighted round0robin: assign request based on predefined server capacity (eg large server get doubt the traffic).

IP HASH:

Use the client Ip address constantly direct the same use to the same server ensuring session persistence.

Dynamic load balancing: dynamic algorithm makes router decision by active checking real- time performs metric and server condition.

How it work: agent or making system constantly access resource ability (CPU, memory,) before shedding traffic

Advantageous: highly adapted and resilience .it prior individual sever overhead and measuring all over all system uptime.

Dis advantageous: more complex to implements and increase higher computational overhead due to contract monitoring.

ALGORITHMS:

Least communication: router new request to the same with the fewest active connector.

Least request time: faster in both active connector and count the server inherently to fewer the fast response server.

Resource-based: use initiated agent to query count-CPU and memory availability on backend nodes before distributed load.

LOAD BALANCER - types:

Load balancing is a distributed system . distribute incoming networks traffic across the multiple servers . this ensure high availability fault tolerance and optimize source utilization . it prevent any single server from becoming a bottleneck , drastically improving over all system respectively response time.

By implementing types

Hardware load balance: physically and properties application 9 EG CITRIX, F5, Networks) installed in the determine the offer exceptional thought between exception and rigid to scale.

Software load balance: application installed a standard hardware or virtual mechanism(eg NGINX,HAPROXY) they are highly flexibleness efficient and easy to update .

Cloud load balancer: managed fully suitable server provided by cloud vendors (eg AWS elastic load balancing, google cloud load balancing) the integrated directly with auto scaling greps).

BY OSI MODEL LAYERS:

Layer 4: (transport layer) route traffic based on IP address TCP/UDP port without interact the packet content it is highly inefficient and incredibility fast.

Layer 7:(application layer) . "smart" nodes that inspect packet content , HTTP header, VPN and COOKIES idea for micro server allowing request to be requested to specific server based on the request context (EG /apiVS/image).

MICRO SERVICE:
it is a type of distributed system while " distributed system " is a broad categories of computing. Micro server are specific implementation where the system application to broken down in to small loosely coupled and independently delegable source that communicate over networks . because micro service are distributed system they introduced complexity that traditional monolithic application do not leave your trade in-memory simplicity for network latency complex and they too like centralization and service method.

CONCLUSION:
distributed computing and decentralized load balancing are deeply inter connected computers by assigning task across a network of independent computer distributed computing achieves its ultimate goal of mixing resource utilization making decentralized load balancing a fundamental design goal rather then just a final archives , because it is independent task allocation instead of relying in a centralized " TRAFFIC COP" nodes independently evaluate their own work load . the decentralized distribution is resource optimization . then the algorithm automatically migrate excess task from heavily loaded processor to lightly loaded or idle access improve execute speed by availability a single focal point for reducing division . they architecture can easily service and adapt to dynamic change in the network. It has fault – tolerance . if a single routing mechanism goes down the rest of the nides continuous to collaborate without held the entire system . because division – making is shared convey all control nodes, it eliminate the check point common traditional centralized setup . it allows network to dynamically adjust to changing work load through it can requires high communication overhead to constant SYNC system state.

REFERENCES:

1..M. Shoeybi, "Megatron-LM: Training multi-billion parameter language models using model parallelism," 2019, arXiv: 1909.08053.

2.J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, arXiv: 1810.04805.

3.A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," 2020, arXiv: 2010.11929.

4.N. Parmar, "Image transformer," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 4055–4064.

5.T. Brown, "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901.

6.R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, no. 1, pp. 79–87, 1991.

7.W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *J. Mach. Learn. Res.*, vol. 23, no. 120, pp. 1–39, 2022.

8.D. Dai, "DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models," 2024, arXiv:2401.06066. [Online]. Available: https://arxiv.org/abs/2401.06066

9.J. He, "Fastermoe: Modeling and optimizing training of large-scale dynamic pre-trained models," in *Proc. 27th ACM SIGPLAN Symp. Princ. Pract. Parallel Program.*, 2022, pp. 120–134.

10. A. Paszke, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035.

11.S. Pal, "Optimizing multi-GPU parallelization strategies for deep learning training," *IEEE Micro*, vol. 39, no. 5, pp. 91–101, Sep./Oct. 2019.

12.S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "Zero: Memory optimizations toward training trillion parameter models," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, 2020, pp. 1–16.